VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozšířená realita

Augmented Reality

2019

Bc. Ondřej Řeháček

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Ondřej Řeháček**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rozšířená realita
Augmented Reality

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je realizovat ukázkové řešení dynamické projekce s využitím snímání tvaru scény, tzv. Augmented reality. Výsledkem práce bude aplikace určená k prezentaci a testování této nové technologické oblasti.

1. Seznamte se s možnostmi snímání scény a vývoje ve vazbě na zařízení umožňující rozšířenou realitu - HoloLens, Mixed Reality by Microsoft.
2. Navrhněte aplikaci, která bude schopna generovat obraz na základě dat získaných externě či díky hardwarovým možnostem zařízení.
3. Realizujte různé formy vizualizace a interakce s důrazem na implementaci prvků rozšířené reality.
4. Zhodnoťte výslednou aplikaci a celkový koncept rozšířené reality.

Seznam doporučené odborné literatury:

[1] S. Ong: Beginning Windows Mixed Reality Programming: For HoloLens and Mixed Reality Headsets, Apress, 2017, ISBN: 978-1484227688
[2] D. Catuhe: Programming with the Kinect for Windows Software Development Kit, Microsoft Press, 2012, ISBN: 978-0735666818
[3] Win2D, http://microsoft.github.io/Win2D/html/Introduction.htm
[4] J. Newnham: Microsoft HoloLens By Example, packt Publishing, 2017, ISBN: 978-1787126268

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2019 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2019 .....

Rád bych na tomto místě poděkoval Ing. Michalu Radeckému,Ph.D za odbornou pomoc a konzultaci při zpracování této diplomové práce. Dále chci poděkovat svým rodičům, kteří mi umožnili studovat a své přítelkyni za nezměrnou podporu a motivaci.

Abstrakt

Diplomová práce se zabývá popisem rozšířené reality (AR). Pokrývá úvod do problematiky tohoto oboru, jeho historii a popisuje samotný koncept AR aplikací. Specializuje se na platformu Windows Mixed Reality společnosti Microsoft a na vývoj v nástroji Unity. Praktická část diplomové práce se skládá z vývoje ukázkových aplikací čerpajících funkcionality specializovaného hardware Microsoft Hololens. Závěrem je hodnocení procesu vývoje a přínosu rozšířené reality pro obor výpočetní techniky a pro širokou veřejnost.

Klíčová slova: Rozšířená realita, Windows Mixed Reality, Vuforia, Unity, Microsoft Hololens

Abstract

This thesis contains a comprehensive introduction into the field of augmented reality (AR), history and concept of AR applications. Theoretical part explores current software platforms, development kits and best practices for proper design and development of applications on one of the major AR platforms today - Windows Mixed Reality using the Unity Engine. The practical part of the thesis consists of the development of sample application that leverages functionality provided by Microsoft Hololens. Work is concluded by an evaluation of the development process and the impact of augmented reality in the industry and on the general public.

Key Words: Augmented reality, Windows Mixed Reality, Vuforia, Unity, Microsoft Hololens

Obsah

Seznam použitých zkratek a symbolů

6DOF	–	6-Degrees Of Freedom
AFRL	–	Air Force Research Laboratory
API	–	Application Programming Interface
AR	–	Augmented Reality
COM	–	Digital Versatile Disc
CPU	–	Central Processing Unit
DSI	–	Display Serial interface
FOV	–	Field Of View
GPS	–	Global Positioning System
GPU	–	Graphics Processing Unit
HMD	–	Head Mounted Display
HPU	–	Holographic Processing Unit
HUD	–	Heads-Up Display
IDE	–	Integrated Development Enviroment
IMU	–	Inertial Measurement Unit
JPEG	–	Joint Photographics Experts Group
LCD	–	Liquid Crystal Display
MR	–	Mixed Reality
PNG	–	Portable Network Graphics
QR	–	Quick Response Code
RGB	–	(Red, Green, Blue) Colorspace
SDK	–	Software Development Kit
SLAM	–	Simultaneous localization and mapping
ToF	–	Time-of-Flight
UX	–	User Experience
VIO	–	Visual odometry
VISLAM	–	Visual Inherited Simultaneous localization and mapping
VR	–	Virtual Reality
WMR	–	Windows Mixed Reality

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V posledních letech zažívá oblast mobilní a nositelné elektroniky obrovský nárůst výkonu a funkcionality. Většina z nás vlastní mobilní zařízení s výkonem téměř srovnatelným s plnohodnotným přenosným počítačem a od toho se odvíjí také poskytovaná funkcionalita a komplexnost aplikací pro každodenní použití. Jedna z oblastí, která se těší velkého rozšíření a popularity na mobilních platformách je právě rozšířená realita. Videohry a sociální aplikace prvky rozšířené reality již aktivně využívají a u uživatelů se těší velké obliby, ať už z hlediska zábavy, tak produktivity.

Dle předního statistického portálu Statista se v roce 2016 hodnota trhu s rozšířenou realitou pohybovala okolo **6 miliard dolarů**. Dnes již přesahuje **20 miliardovou hranici** a předpokládá se růst trhu až na astronomických **192 miliard dolarů do roku 2025** [1]. Není tedy žádným překvapením, že vzhledem k potenciálu oboru většina velkých hráčů na poli výpočetní techniky investuje každým rokem více času a financí právě do hluboké integrace rozšířené reality do jejich produktů a nabízí své proprietární řešení pro vývoj aplikací. Je pouhou otázkou času, než se rozšířená realita stane nepostradatelným nástrojem pro zábavu i každodenní práci milionů lidí po celém světě.

Teoretická část této diplomové práce se primárně zaměří na představení samotného konceptu rozšířené reality, historie jeho využití a nasazení této technologie na moderní zařízení a platformy. Popíšeme si aktuální trendy a hardware v tomto odvětví, hlouběji se pak zaměříme na vývoj a návrh aplikací pro platformu Windows Mixed Reality společnosti Microsoft a na vývoj pro specializovaný hardware pro rozšířenou realitu - Microsoft Hololens. Popíšeme si nastavení a užití vývojového prostředí Unity, správné postupy při návrhu aplikací, vývoj pomocí balíku Mixed Reality Toolkit (MRTK) a rozpoznávání objektů a obrázků, pomocí předního frameworku Vuforia.

Vzhledem k tomu, že práce spadá do zaměření analýzy a zpracování dat, praktická část se zaměří na několik možnosti vizualizace dat pomocí rozšířené reality s obohacením o funkcionalitu poskytovanou zařízením Hololens - Gesta rukou, snímání scény a detekce 2D a 3D kotevních bodů. Vizualizace rozsáhlých dat může být limitována 2D zobrazovacím zařízením a rozšířená realita by v budoucnu mohla představovat neocenitelný nástroj pro obor analýzy dat.

2 Rozšířená realita

Rozšířená realita (Augmented Reality / zkráceně "AR") je technologie využívaná k doplnění reálného obrazu (z kamerového zdroje či průhledného skla) v reálném čase o počítačově generovaný text, video či trojrozměrné modely. Na rozdíl od virtuální reality není zde prostředí kompletně generováno výpočetním zařízením, ale reálný svět je pouze obohacen o další informace a interaktivní 3D modely. [6]

Obrázek 1: Model loga katedry v AR aplikaci

V dnešní době AR není pouze záležitostí komplexních a specializovaných zařízení užívaných pro vojenské či lékařské účely jako tomu bylo dříve. Dnes nalezneme aplikaci i v dostupných přenosných mobilních zařízení či specializované technice. Kamera, která je připojená k zařízení nebo kamera v mobilním telefonu, snímá obraz reality společně s dalšími senzory detekující hloubku vstupního obrazu (depth camera), orientaci (gyroskop) a přesnou lokaci (IMU, Wi-Fi, GPS). [6] Specializovaný software a hardware poté tyto vstupy analyzuje v reálném čase a doplňuje ho vzhledem k informacím ze senzorů o dodatečný výstup. Výpočetní náročnost rozšířené reality je poměrně vysoká a nasazení v minulosti bylo tedy velmi finančně náročné. [6]

2.1 Dělení AR aplikací

Samotný koncept rozšířené reality pokrývá velmi široké spektrum rozdílných aplikací a reprezentuje spolupráci velkého počtu algoritmů a provedeného výzkumu z různých oblasti strojového vidění, zpracování dat, počítačové grafiky a robotiky. Aplikace pro rozšířenou realitu je tedy

vhodné rozdělit do několika základních kategorií dle jejich komplexnosti a nabízené funkcionality. [7]

- AR aplikace založené na jednoduchých kotevních bodech **(Marker-based)**.
- AR aplikace bez kotevních bodů **(Markerless)**.
- AR aplikace bez kotevních bodů s porozuměním geometrických tvarů. **(Markerless with geometric environment understanding)**
- AR aplikace bez kotevních bodů s porozuměním okolí a určení sémantiky. **(Markerless with spatial understanding)**

Samozřejmě nalezneme aplikace, které jednotlivé funkcionality kombinují a není je tedy možné specificky zařadit do některé z jmenovaných kategorii.

AR aplikace založené na jednoduchých kotevních bodech jsou především implementace, které reprezentují detekci objektů v obraze ať už ve fotografii, videu nebo v reálném čase pomocí fotoaparátu. Jedná se tedy o zobrazení generovaného obsahu na předem definované kotevní místo – unikátní rozlišovací tvar, QR kód či obrázek. Generované objekty se poté k takovému kotevnímu bodu přiřadí a pohybují se současně s ním v reálném čase. [7]

AR aplikace bez kotevních bodů jsou aplikace, které na rozdíl od předchozí kategorie nevyžadují umístění kotevního bodu; ve většině případů se tedy jedná o umístění levitujícího 3D objektu do obrazu kamery s možností objekt prohlížet z různých stran v závislosti na pohybu zobrazovacího zařízení. [7]

AR aplikace bez kotevních bodů s porozuměním tvarů. V tomto případě jsou AR aplikace obohaceny o 3D rekonstrukci okolí zařízení za pomocí rozšířených algoritmů a dodatečného hardware (hloubková kamera). Díky tomu dokáže 3D obsah reagovat na objekty v prostředí – umístění objektu, nastavení limitu. [7]

AR aplikace bez kotevních bodů s porozuměním tvarů a sémantiky. V tomto případě jsou AR aplikace obohaceny nejen o přesnou 3D rekonstrukci, ale za pomoci dodatečných algoritmu dokážou i jednotlivé geometrické tvary v okolí zařadit do kategorií, a tudíž rozpoznat, zda se jedná například o stůl, židli, lavičku a další. Tato schopnost umožňuje ještě mnohem hlubší provázanost generovaného obsahu s reálným světem. [7]

2.2 Požadavky na hardware

Obecně platí, že čím více kvalitních senzorů a výpočetního výkonu zobrazovací zařízení má, o to více můžeme zdokonalit zážitek z AR aplikací. Mimo výkonný výpočetní procesor, který dokáže dostatečně rychle zpracovávat vstupní data, sem patří také set periferii pro základní AR: [2] [6]

- **Akcelerometr** - Detekce orientace.
- **Gyroskop** - Detekce naklonění a natočení.
- **RGB Kamera** - Vstupní obraz / standardní barevná digitální kamera.

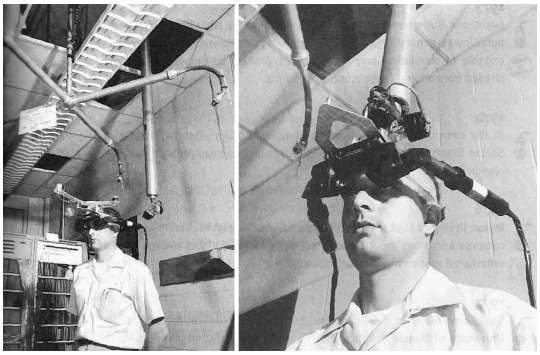
Akcelerometr a Gyroskop jsou v modernějších zařízeních nahrazeny **IMU jednotkou (inertial measurement unit)** - jenž je kombinace několika akcelerometru, gyroskopů a magnetometrů. Jedná se o extrémně přesné zařízení pro určení pohybu v prostředí, kde není dostupný GPS signál. Nalezneme však spoustu dalších periferii, které mohou AR podstatně obohatit a vytvořit mnohem příjemnější zážitek. [2]

- **Time-Of-Flight (ToF) kamera** - Hloubková (3D) kamera.
- **Světelný senzor** - Měření okolního světla (light estimation).
- **Vícekanálový zvuk** - Obohacení obsahu o prostorový zvuk (spatial sound).
- **Mikrofon** - Reakce obsahu na okolní zvuk.
- **GPS** - Pro určení kotevních bodů dle lokace (world tracking, beacons).

V dnešní době je z hlediska cenové dostupnosti a rozšířenosti nejlepším zařízením pro AR smartphone či tablet. Proto se většina vývoje v oblasti AR specializuje právě na tyto zařízení a hlubokou implementaci do mobilních operačních systému (Android, iOS). Druhou kategorií jsou poté dedikované zařízení vyvinuté speciálně pro konzumaci AR obsahu (Microsoft Hololens, Magic Leap Lightwear, Vuzix Blade AR, Garmin Varia Vision a ODG R-7) [1] [6]

2.3 Historie

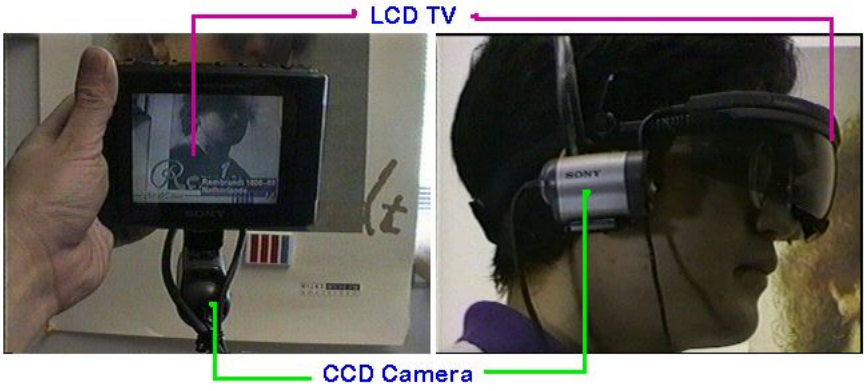
Historie vývoje [5] virtuální a rozšířené reality sahá až do roku 1968. Tehdy americký vědec a uznávaná osoba v oblasti moderní počítačové grafiky, **Ivan Sutherland**, při svém působení na **Harvard University** vytvořil historicky první set pro virtuální a rozšířenou realitu. Jednalo se o průhledné brýle, do kterých mohla být vykreslená vektorová grafika. Zobrazení taktéž reagovalo na pohyb uživatele pomocí mechanických a ultrasonických senzorů pohybu. [5]



Obrázek 2: System pro VR/AR, Ivan Sutherland (1968)

V průběhu 90.let byl výpočetní výkon zařízení již dostatečný pro realizaci použitelné rozšířené reality. V roce 1992 **Tom Caudell a David Mizell** stáli za definici pojmu rozšířené reality a od té doby se termín začal oficiálně užívat. [5]

V průběhu roku 1994 došlo k představení projektu **"NaviCam"** [1] v Japonsku, který lze považovat za první koncept marker-based AR aplikace. **Jun Rekimoto** a **Katashi Nagao** představili koncept skenování značek z video zdroje a obohacení videa na základě informací obsažených ve značce. Představen byl také koncept **QR kódu** (společnost Denso Wave) a dalších různorodých grafických markerů pro strojové vidění a automatizaci. [5]



Obrázek 3: Systém NaviCam (1994)

Ve stejném roce společnost Sony uvedla na trh **"Glasstron"** [2], kompaktnější průhledný HMD dostupný pro širokou veřejnost. Společnost Sony také demonstrovala první aplikace AR ve videohrách na jejich herní konzoli Playstation. [5]

V roce 1999 **Hirokazu Kato** a **Mark Billinghurst** vydali **ARToolKit**, první open-source softwarový framework pro vývoj marker-based AR, který je dodnes aktivně vyvíjen a používán v mnoha projektech. [3]

Od roku 2000 do roku 2003 byl už vývoj AR aplikaci na všech frontách v plném pohybu, došlo k představení spousty proprietárních AR systému, ať už stacionárních, tak mobilních, pro navigaci, lékařské, vojenské nasazení a pro širokou veřejnost v zábavních parcích či video herních klubech. [5]

V roce 2002 byl na trh uveden první smartphone **Nokia 7650**. V té době s přelomovým mobilním operačním systémem **Symbian**. Svou formou a vybaveností už připomínal koncept moderních chytrých telefonu. Krátce po zahájení prodeje **Mathias Mohring** představil koncept marker-based mobilních AR aplikací právě na Nokia 7650. Jednalo se o první demonstraci AR na veřejně dostupném mobilním zařízení. Následující rok se na trh chytrých telefonů dostává společnost Siemens s modelem **SX1**, který je dodávaný s AR videohrou **Mozzies - "Mosquito Hunt"**. [5] [1]

Obrázek 4: ARToolkit na Symbian a Palm OS

Na trhu se paralelně začal v roce 2007 objevovat **Apple iPhone** a první telefony založené na **OS Android**. Započaly novou éru mobilních zařízení, které se později staly primární platformou pro dnešní AR aplikace.

[1] https://www2.sonycsl.co.jp/person/rekimoto/navi.html
[2] http://www.steves-digicams.com/glasstron.html
[3] http://www.hitl.washington.edu/artoolkit/documentation/history.htm

V roce 2009 společnost Microsoft prezentovala **"Project Natal"** pro herní konzoli Xbox 360. Jednalo se hloubkovou kameru, pomocí které lze ovládat video herní obsah pouze pomocí gest a pohybu těla uživatele. Project Natal se později přejmenoval na **"Kinect"** [2] a dostal se do veřejného prodeje koncem roku 2010. Položil základy pro celý mixed reality ekosystém společnosti Microsoft - Microsoft Hololens a Azure Kinect. Výrobce mobilních chipsetu Qualcomm začal vážně investovat do vývoje AR aplikací a v roce 2011 prezentoval svou otevřenou AR platformu zvanou **"QCAR - Qualcomm Augmented Reality"**. Platforma se později přejmenovala na **"Vuforia"** [3] a dodnes je jedním z nejzásadnějších a nejkomplexnějších AR vývojových nástrojů. [5]

Společnost Apple provedla v roce 2013 akvizici společnosti **Primesense** (která se podílela na vývoji Microsoft Kinect) a dala tím najevo, že hodlá silně investovat do AR vývoje a výzkumu. V roce 2015 Microsoft představil první koncepty projektu **Baraboo - Hololens**. První development kit se dostal do prodeje v roce 2016. [5]

[1] https://shkspr.mobi/blog/tag/mozzies/
[2] https://www.jameco.com/jameco/workshop/howitworks/xboxkinect.html
[3] https://www.ptc.com/en/products/augmented-reality

3 AR platformy

S rapidním růstem výkonu mobilních zařízení, výrazným zlepšením kvality integrovaných kamer a cenové přístupnosti široké škály přesných senzorů se v posledních letech výrazně rozšířil trh se specializovaným hardwarem pro AR. Otevřel se však také nový svět softwarových balíku (SDK) jak komerčních, tak open-source, které umožňují integraci AR funkcionality do již existujících mobilních zařízení.

Nyní si představíme nějaké z dostupných API pro efektivní vývoj AR aplikaci na různorodých platformách. Většina vyjmenovaných API reprezentují základní komponentu operačního systému, umožňující pohodlnější nativní vývoj aplikací pro AR. **Primárně se budeme věnovat platformě Windows Mixed Reality a frameworku Vuforia, pomocí kterého je vyvinutá aplikace v praktické části diplomové práce.**

3.1 Windows Mixed Reality

Microsoft pro svůj OS Windows 10 připravil specializovanou platformu pro vývoj aplikací pro rozšířenou realitu [1]. Pokrývá jak vývoj AR a VR aplikací pro stávající zařízení běžící na systému Windows 10 (x86-64 i ARM), tak pro specializovaný hardware, který společnost Microsoft nabízí. WMR byl implementován jako součást jádra Windows společně se spuštěním veřejného prodeje Microsoft Hololens DK1 v roce 2016. [8]

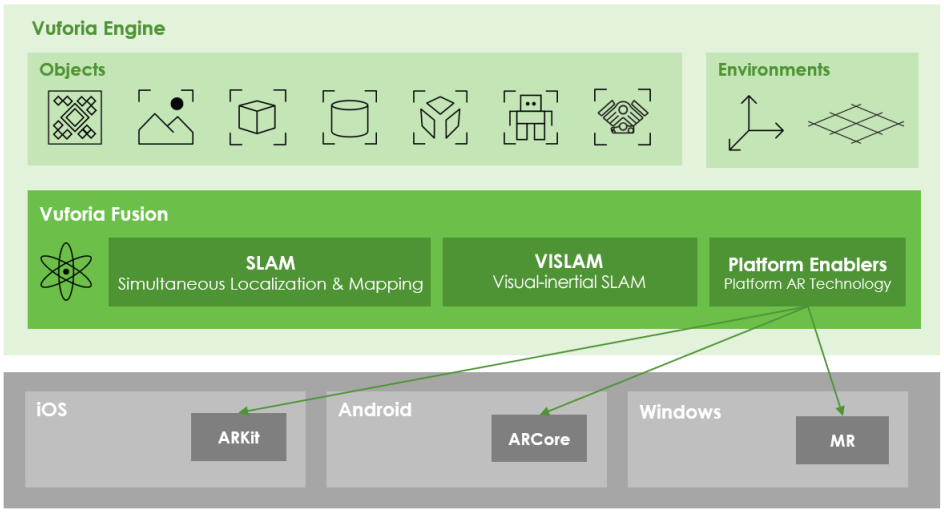
V průběhu roku 2017 byly do rodiny WMR přidány cenově dostupnější **"Immersive Headsety"** od partnerů Acer, Asus, HP, Lenovo, Samsung a DELL. Zážitek z AR je však na těchto zařízeních podstatně omezen a je vhodnější zařízení označovat spíše jako cenově dostupné VR headsety. [2]

Windows Mixed Reality nabízí možnost provozu nejen standardních "Full-AR"zážitků implementovaných v nástrojích jako je Unity, ale taktéž umožňuje spouštění klasických UWP aplikací "v okně", které mají prostředí zpracováno v XAML, stejně jako kterákoliv jiná moderní .NET aplikace pro desktopovou verzi Windows. Počítá se tedy s možností multitaskingu a WMR tak představuje svou implementaci jakousi upravenou grafickou nadstavbu pro přizpůsobení plnohodnotného OS Windows při běhu na AR zobrazovací jednotce. V případě implementace WMR je popsat konkretní řešení poměrně komplikované. Jedná se o proprietární modifikaci standardních **VO (Visual Odometry)** algoritmů.

[1] https://developer.microsoft.com/en-us/mixed-reality

3.2 Vuforia

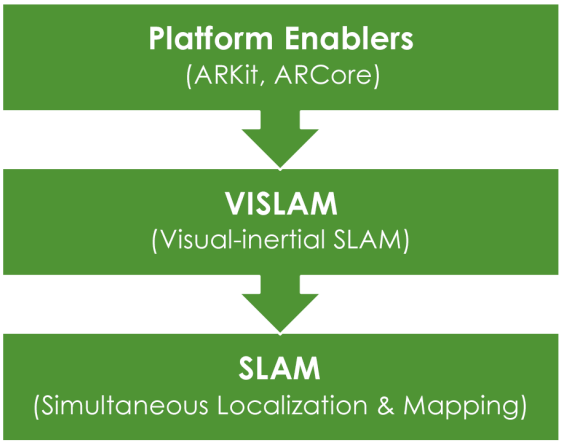
Vuforia [10] je aktuálně nejkomplexnějším a nejdéle vyvíjeným softwarovým balíkem pro vývoj aplikací pro rozšířenou realitu. Pro platformu vuforia lze vyvíjet v širokém spektru IDE:Unity, Android Studio, Xcode a Visual Studio, a tedy i pomocí široké škály programovacích jazyků: C++, Objective C, C#, Java. Výsledné aplikace mohou běžet jak na platformě Windows, tak na iOS a Android. Pokud je zařízení, na kterém je aplikace spuštěna hardwarově kompatibilní s výše zmíněnými API (ARkit, ARcore, WMR), Vuforia automaticky tyto API využívá pro maximální optimalizaci běhu aplikace, v opačném případě použije svou vlastní softwarovou implementaci a engine. Tento fakt umožňuje Vuforia aplikacím běžet ještě na mnohem bohatší škále zařízení, a přesto nepředstavuje žádné výrazné limitace oproti ARCore či ARKit na podporovaném moderním hardware. Vuforia nabízí unikátní funkcionalitu jako je například **Model Targeting with Deep Learning** - implementace Deep Learningu pro automatické rozpoznávání i sofistikovanějších reálných objektů pouze z aproximačního vstupního 3D modelu. **Image a Multi Targets** - Efektivní rozpoznávání vícero 2D objektu a QR kódu. **Cylindrical mapping** pro mapování textur a obsahu na reálné objekty cylindrického tvaru.



Obrázek 5: Komponenty Vuforia Engine [9]

Snad nejdůležitějším komponentem frameworku je **Vuforia Fusion** - [9] Nástroj, jehož cílem je řešit vysokou fragmentaci platforem v oblasti AR. Jedná se o middleware, pomocí kterého lze nasadit jednu aplikaci na nejširší portfolio zařízení s rozdílnými nativními API - ARKit, ARCore a Windows MR. Fusion je tedy "překladačem", mezi nativním API a samotným Vuforia Engine

a umožní nasazení jedné aplikace na tři různé platformy a navíc zajistí zachování funkčnosti sledování zařízení v prostoru a alespoň jedné ze dvou základních AR komponent - Ground Plane a Model Targets i na starších (nativním API nepodporovaném) zařízení pomocí volitelné softwarové implementace některého z algoritmu pro mapování prostředí. Fusion se skládá ze 3 rozdílných komponentů, které jsou řazeny hierarchicky a jsou zvoleny pro zpracování AR obsahu na základě kompatibility a volných HW prostředcích zařízení, na kterém aplikace běží.



Obrázek 6: Hierarchie komponent Vuforia Fusion [9]

- **Platform Enabler -** Je využit v případě, že se jedná o moderní zařízení a podporuje některý z nativních API - Využívá se algoritmu zpracování AR scény implementovaných v samotném systémovém API. *(například iPhone 6S až X/XS)* [1].
- **VISLAM -** Pokud zařízení není podporováno některým ze 3 nativních API, je využita další možnost v hierarchii Vuforia Fusion - softwarová implementace algoritmu VISLAM (Visual-inertial SLAM) - kombinace algoritmu VIO a SLAM který zachovává Ground Plane Tracking i Model Targets *(například iPhone 6)* [1].
- **SLAM -** V případě, že zařízení nemá dostatečné prostředky ani pro zpracování VISLAM, je zvolena poslední "fallback"možnost - méně náročnější implementace algoritmu SLAM, který zachová funkční alespoň Ground Plane Tracking pro základní AR aplikace *(například iPhone 5 a 5S)* [1].

[1] https://library.vuforia.com/articles/Solution/vuforia-fusion-supported-devices.html

3.3 Apple ARKit

API mobilního operačního systému iOS [1], které umožňuje vývojářům efektivní vývoj AR aplikací pro iPhone a iPad. Apple naprosto radikálně projevil zájem o hlubší integraci AR do svých produktů, jak nasazením dedikovaného API, ale také obohacením nových produktů o důležité hardwarové komponenty – optimalizace procesoru (A11/A12 Bionic) specificky pro AR operace. Aplikace pro ARkit lze vyvíjet pomocí jazyka **Objective-C** a alternativě pomocí jazyka **Swift**. Standardně obsahuje Sledování 3D i 2D objektů v prostoru, detekce jak horizontálních, tak vertikálních a šikmých povrchů, detekce obrázků a QR kódu (plane detection a hit-testing). Specifickou vlastností je pak iBeacons + World Tracking, který umožní jednotlivé AR objekty ukotvit na kterékoliv místo na světě. [11]

3.4 Google ARCore

Stejně jako Apple, i další velký hráč na trhu s mobilním zařízeními, Google, nabízí SDK s názvem ARcore [2]. ARcore je multiplatformní a lze ho využít pro vývoj jak na OS Android, tak na Apple iOS. Aplikace pro ARcore lze vyvíjet stejně jako všechna řešení v Google ekosystému v programovacím jazyce **Kotlin** či **Java**. ARcore dokáže sledovat pozici zařízení na **6 osách (takzvaný 6-degrees of freedom)** ve vztahu k okolnímu prostředí za pomocí technologie **COM (Concurrent Odometry)**. Implementována je také velmi robustní detekce jak horizontálních, tak vertikální a šikmých povrchů. Díky tomu aplikace dokáže spolehlivě detekovat objekty reálného světa, jako je zem, zdi, stoly, židle, lavičky pro lepší porozumění okolního prostředí. Framework staví na 3 základních principech - **detekce pohybu (Motion Tracking), porozumění okolí (Enviromental Understanding) a detekce osvětlení okolí (Light Estimation)**. [11] [12]

[1] https://developer.apple.com/arkit/
[2] https://developers.google.com/ar/discover/

3.5 ARToolKit

Z historického hlediska je pravděpodobně jedním z nejrobustnějším a nejdéle vyvíjeným řešením ARToolKit [1]. Jedná se o více než 20 let vyvíjený framework, který se dá považovat za produkt který vybudoval cestu k moderním AR řešení. Svými schopnostmi již dnes bohužel nedosahuje funkcionality výše zmíněných komerčních řešení, přesto představuje velmi robustní multiplatformní implementaci pro základní marker-based AR aplikace zcela zdarma. Již od roku 1999 byl používán pro počítačové AR aplikace a v roce 2005 byl integrován jako první mobilní AR SDK do operačního systému Symbian pro smartphony společnosti Nokia a Siemens. Později v letech 2009-2011 byl taktéž pilotním řešením pro dnešní moderní operační systémy - iOS a Android, o mnoho let dříve, než byly představeny jejich nativní AR API. [13]

3.5.1 AR.js / ARToolKit.js

Jedním z řešení založených na dlouhodobém vývoji artoolkit je právě AR.js [2]. Jedná se o robustní framework umožňující AR obsah integrovat přímo do webových prezentací. Právě díky javascriptovým a WebGL frameworkům jako je AR.JS lze implementovat aplikace rozšířené reality i na zařízeních, které nemají žádná dostupná řešení, ať už se jedná o starší nepodporovaná mobilní zařízení nebo o zařízení s alternativními operačními systémy. Aplikace mohou fungovat na každém zařízení, které podporuje některý z moderních webových prohlížečů a je vybaveno fotoaparátem a alespoň základní skupinou pohybových senzorů.

[1] https://github.com/artoolkit
[2] https://github.com/jeromeetienne/AR.js/

3.6 Srovnání

Framework	Licence	Programovací jazyky	Platformy
WMR	Komerční	Sharp, C++	Windows 10
ARkit	Komerční	Objective-C, Swift	iOS
ARCore	Open-Source	Java, Kotlin	Android, iOS
Vuforia	Komerční	Sharp, C++	Android, iOS a různé HMD
ARToolKit	Open-Source	C++	Windows, Linux, iOS, Android
AR.js	Open-Source	JavaScript	WebGL

Framework	Kotevní body	Detekce Ploch	Mapování okolí	Komplexnější kotevní body	Detekce osvětlení
WMR	**X**	**X**	**X**		
ARkit	**X**	**X**			**X**
ARCore	**X**	**X**	**X**		**X**
Vuforia	**X**	**X**		**X**	
ARToolKit	**X**				
AR.js	**X**				

4 Vývoj pro Windows Mixed Reality

Následující kapitola popisuje proces návrhu a vývoje aplikace pro **Windows Mixed Reality**, který budeme využívat při implementaci výsledné aplikace. Ukážeme si popis ověřených postupů při návrhu aplikace, popíšeme základy vývoje AR aplikací v prostředí **Unity**, využití **MRTK** pro implementaci snímání tvaru scény a frameworku **Vuforia** pro vytvoření kotevních bodů v naší aplikaci. [2]

4.1 Návrh AR aplikací

AR aplikace představují kompletně jiná paradigma návrhu oproti klasické 2D aplikaci běžící na displayi mobilního zařízení či monitoru osobního počítače. Tyto změny jsou výrazné jak z pohledu návrhu aplikací, tak při samotném postupu při vývoji. [14]

4.1.1 Základní funkcionalita

Samotná aplikace by měla využívat alespoň jednoho z druhů detekce ploch okolí. Pokud aplikace dokáže reagovat na vícero ploch v okolí uživatele, uživatel získá o to lepší AR zážitek. Navíc lze také implementovat reakci na osvětlení či ozvučení okolí, implementaci stínu v reálném čase, či hlubší porozumění reálných objektů v okolí (pro rozdílné reakce AR objektů ke vztahu k objektům reálným). Implementace různých kotevních bodů a rozpoznávání vzorů je také velmi silným nástrojem k prohloubení realismu v aplikaci. [14]

4.1.2 Respektovat omezení aktuálního hardware

V případě, že uživatel nevlastní žádný z dedikovaných AR headsetů, lze předpokládat, že AR obsah bude konzumovat prostřednictvím mobilního zařízení a na obsah bude nahlížet pomoci LCD displeje. Toto představuje limitaci převážně z hlediska ovládání. Uživatel jednou rukou drží samotné zařízení, a tedy práce s aplikací bude omezena pouze na interakci jednou rukou. V našem případě budeme vyvíjet pro zařízení Hololens, nebudeme sice limitování ovládáním, ale je třeba uvažovat nad snímkovací frekvencí zobrazovacího zařízení a dle toho přizpůsobit rychlost animací v samotné aplikaci, případně šetřit výpočetním výkonem zařízení, který je stále u Hololens silně limitován. [14]

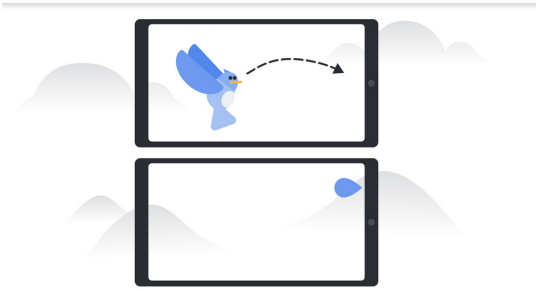
4.1.3 Trojrozměrná responzivita

Je důležité si uvědomit, že samotná AR aplikace neexistuje v rámci 2D plochy jako je monitor či display zařízení, ale všude okolo uživatele. Zařízení funguje pouze jako okno k nahlížení do AR světa. Žádný ze zajetých postupů návrhu UI pro desktopové a mobilní aplikace zde nefunguje. Za nejbližším návrhu AR aplikací můžeme přirovnáním považovat koncept responzivního webu, kde počítáme s tím, že webová prezentace bude zobrazena na zařízeních s různými vlastnostmi a velikosti displeje. Zde se ale nejedná o responzivitu ve 2D, ale ve 3D. Jedna aplikace se bude

muset umět přizpůsobit velkému množství různorodých prostředí, ve kterém se nachází uživatel. Prototypování by tudíž mělo probíhat ve stylu skicování samotného okolí uživatele, nikoliv jako zobrazení v určitém zařízení. [14]

4.1.4 Práce s prostorem

Při návrhu AR aplikací, je třeba ignorovat klasický koncept oken a nabídek. Pracujeme s 360-stupňovým prostředím, a proto je zcela nevhodné všechny AR objekty umístit přímo před uživatele. Je zcela v pořádku umístit objekty i mimo viditelné okno (viewport) AR prostoru. Naopak takovéto umístění nutí uživatele více se pohybovat uvnitř AR aplikace, což vede k více pohlcujícímu zážitku. [14]



Obrázek 7: AR objekt opouští viditelný prostor a indikační šipka motivuje uživatele k pohybu. [14]

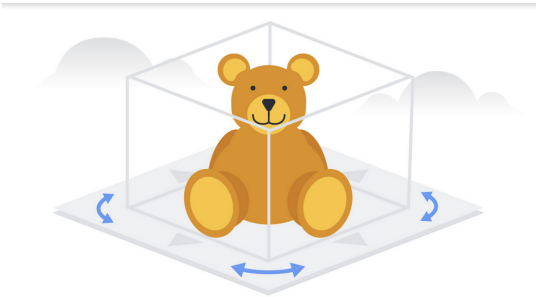
4.1.5 Velikost AR světa a jeho objektů

AR aplikace by měla mít určenou rozlohu generovaného obsahu a jednotlivých objektů v relaci k uživateli aplikace. Jasně bychom si měli definovat, zda je obsah určen pro jednu plochu **(Seated/Standing Scale)**, či je velikostně vhodný do celého pokoje **(Room Scale)**, nebo je tak rozsáhlý, že je naopak ideální do mnohem větších venkovních prostoru **(World Scale)**. Není vhodné mezi těmito kategoriemi přecházet nebo se snažit aplikaci implementovat do více rozdílných prostorů. Uživatel by měl být o potřebě rozlohy okolí informován před užíváním aplikace, aby nedošlo k obstrukci AR prvků. [14]

Obrázek 8: Špatně specifikovaná velikost aplikace může vést k nepoužitelnosti aplikace. [14]

4.1.6 Interakce s AR objekty a jejich vlastnosti

V momentě, kdy máme jasně definovanou funkcionalitu a rozsah naší aplikace, je nutností se zamyslet také nad interakci a manipulaci s jednotlivými objekty v AR. Důležitým prvkem interakce s objekty je implementace zpětné vazby v případě kolize s AR objektem, ať už vizuální (efekty kamery, vizuální efekty) či zvukové (výstražný zvuk). V případě manipulace je vhodné objekty, se kterými uživatel pracuje, označit kontrolním boxem (tzv. Bounding Boxem či Reticle Selection), který pomáhá pomocí gest objekt přesouvat, zvětšovat a rotovat v prostoru. Při přesunu objektu je vhodná implementace zpětné vazby v relaci s okolními plochami (surface feedback), který pod objektem prezentuje vizuální reprezentaci (stín), který označuje bod dopadu objektu v případě upuštění při interakci. [14] [2]



Obrázek 9: Bounding box po stranách AR objektu představuje flexibilní manipulační nástroj. [14]

4.2 Obecné předpoklady

Univerzální AR aplikace pro WMR mohou být vyvíjeny v několika programovacích jazycích podporovaných **frameworkem .NET**. Každý z těchto jazyků přináší benefity a hodí se na vývoj různých typů projektů. Pokud se jedná o projekt, který je graficky či výpočetně velmi náročný, a vyžadujeme maximální užití výkonu zařízení, lze vyvíjet aplikace pro Hololens nativně v **C++ v kombinaci s API DirectX**. [2]

Ve většině případů se však WMR aplikace vytváří v herním enginu **Unity**, který poskytuje možnosti vývoje pro 2D i 3D hry libovolného žánru a zaměření. Kromě grafického prostředí pro tvorbu obsahu podporuje také tvorbu skriptů, především v jazyce **C#** a **UnityScript** (podobná syntaxe jako Javascript). Společnost Microsoft nabízí v oficiálním SDK řadu ukázek a tutoriálu pro Unity i nativní vývoj v C++. [2]

4.2.1 Základní předpoklady pro vývoj [2]

- **PC s aktuálním OS Windows 10** - Alespoň 4-jádrový 64bitový procesor třídy i5/i7, minimálně 8GB RAM a grafickou kartou s podporou API DirectX 11 a výše.

- **Visual Studio 2017 s aktuálním UWP SDK** - Vzhledem k tomu, že se jedná o UWP, je vývojové prostředí identické pro všechny Windows 10 aplikace.
- **Unity** - Herní engine Unity je neocenitelným pomocníkem pro jednoduchý vývoj aplikaci pro AR. V aktuálních verzích již plně integruje nástroje a komponenty pro rozšířenou realitu a UWP.
- (Volitelné) **Hololens Emulator, MRTK** - Obě komponenty výrazně usnadní vývoj a testování aplikací. Emulátor má výraznější nároky na hardware a vyžaduje funkční prostředí Hyper-V (pouze Pro verze Windows 10). Mixed Reality Toolkit je rozšíření pro prostředí Unity a nabídne jednoduché přednastavení scén, kamery a práci se specifickou funkcionalitou Hololens (např. mapování prostředí)

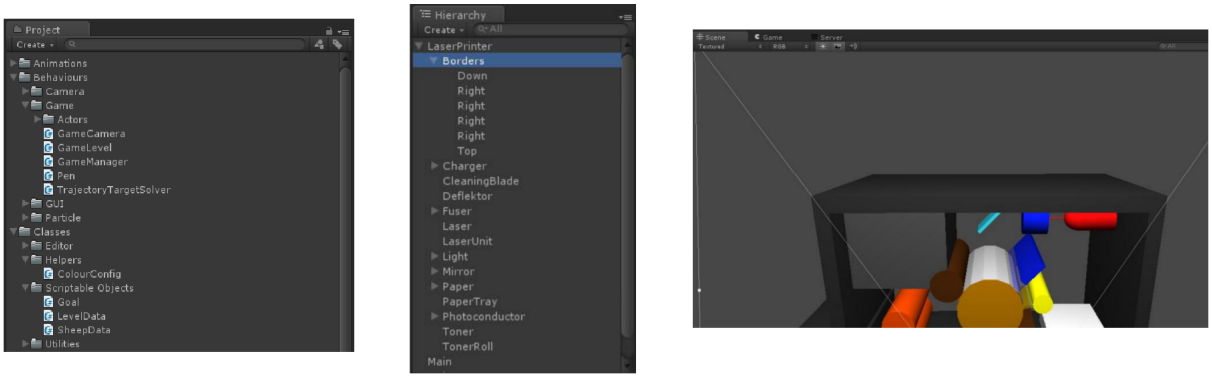
4.3 Editor Unity

Unity je celosvětově označován za přední nástroj pro vývoj videoher pro PC, konzole, mobily a web a nyní také pro vývoj AR aplikací. Práce s projektem se skládá ze dvou částí. Návrh 3D prostředí a implementace jednotlivých objektů. Samotný návrh a náhled scény, editace základních parametrů objektů a přiřazení skriptů se provádí v editoru Unity. Programování, psaní skriptů a ladění naopak v IDE Visual Studio. [4] [15]

4.3.1 Prostředí

Nastavení projektu není třeba nijak editovat, pouze zvolit možnost "3D Project" a všechny potřebné změny nastavit později přímo v editoru po založení projektu. Po nastartování prostředí editoru je pracovní plocha poměrně přehledné rozdělena do 4 hlavních komponent: [4] [15]

- **Inspektor** je panel, pomocí kterého lze upravovat atributy aktuálně označených objektů ve scéně. Jejich pozici, tvar, přiřazovat skripty a mnoho dalšího.
- **Panel hierarchie objektů**, který zobrazuje všechny aktuální objekty ve scéně s jejich hierarchickým řazením.
- **Panel projektu**, ve kterém najdeme veškerý obsah našeho projektu (prefabs, textury, audio soubory, skripty)
- **Náhled na aktivní scénu**, který zobrazuje v několika režimech aktuální rozložení objektů a pohled kamery do zvolené scény.



Obrázek 10: Struktura projektu, hierarchie objektů a náhled na scénu v Unity

4.3.2 Projekt

Všimněme si automaticky vygenerované struktury našeho projektu (v projektovém panelu). Všechny důležité zdroje jsou automaticky ukládány do složky “Assets” tato složka obsahuje samotné scény, skripty, 3D modely, textury, zvukové soubory a další. Manipulace a úprava této struktury je samozřejmě možná, není však doporučena. Chceme-li přidat soubor do projektu, stačí obyčejné přetažení souboru z průzkumníka souboru do editoru, případně za pomocí kontextové nabídky lze vytvořit soubory nové, podobně jako ve většině ostatních IDE. [4]

4.3.3 Prefabs

Důležitou součástí Unity projektu jsou "prefabricated objects", neboli objekty **Prefab** (standardně ukládány do složky "Prefabs") , které označují objekt, ať už se jedná o zvuk, model či jakoukoliv entitu, která nabývá specificky nastavených atributů a přiřazených komponent (například skriptů). Prefab tedy nenese pouze samotný objekt, ale také všechny jeho komponenty a nastavení těchto komponent. Můžeme říct, že se jedná o jakousi standardizaci (lze si představit jako třídu v OOP), pomocí které lze poté vytvářet jednu či více instancí daného objektu, který nabude jednotných atributů a poté tyto instance již pouze modifikovat dle potřeby. [4]

4.3.4 Scéna a její hierarchie objektů

Scénou rozumíme prostor, do kterého umísťujeme jednotlivé prvky - objekty (včetně kamery). V panelu scény určuje a manipulujeme s jejich pozici v prostoru. V panelu hierarchie objektů vidíme jednotlivé samostatné objekty či prefaby v aktuální scéně. Hierarchie je zde důležitá z toho pohledu, že objekty mohou přebírat transformační atributy – pozici, rotaci a zvetšení nadřazených (parent) objektů. [4]

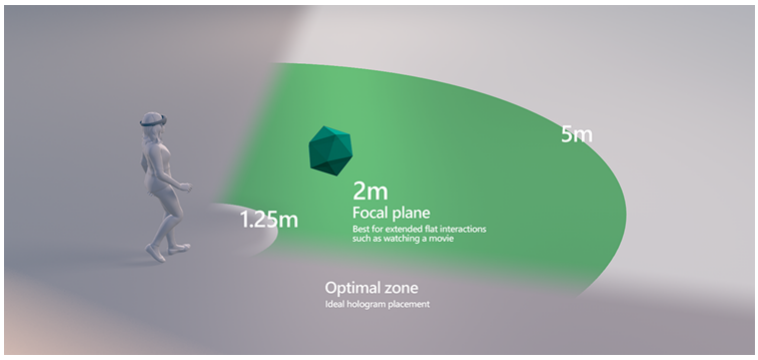
4.4 Mixed Reality Tooolkit (MRTK)

Mixed Reality Toolkit [1] je kolekce komponentů dodávána společností Microsoft, pro standardizaci vývoje pro Windows Mixed Reality. Jedná se o naprosto základní stavební kámen pro efektivní vývoj aplikací pro platformu WMR v Unity. Balík obsahuje součásti pro využívání vstupních zařízení (včetně rozpoznávání gest, pohybových ovladačů), prostorového mapování a zvuku (spatial mapping and sound), standardizované UX ovládací prvky (gaze kurzory, virtuální klávesnice, UI prvky). Jednotlivé komponenty tohoto balíku lze importovat jednoduše do našeho projektu dle potřeby. [16] [17]

4.4.1 Kamera

V případě AR headsetu se stav hardware, jeho pozice a úhel pohledu mapuje k pozici hlavní kamery v Unity. Základní objekt **MixedRealityPlayspace** [2] je označen jako rodič objektu Main Camera. [8] Při založení mixed reality projektu je tedy potřeba pouze vybrat objekty "hlavní kamery"v hierarchii scény a nastavit její na (X: 0, Y: 0, Z: 0) a označit objekt kamery jako potomka objektu**MixedRealityPlayspace**.

Důležitým aspektem je také nastavení **omezení renderování** (near and far cliping planes) AR obsahu z pohledu vzdálenosti od uživatele. Pro stanovení optimální vzdálenosti obsahu od pozice uživatele. Bez omezení muže být renderování jednotlivých objektu příliš blízko či příliš daleko od pozice uživatele velmi nepohodlné a může způsobovat nevolnost, bolest očí (šilhání) či nečitelnost obsahu. Optimální "komfortní"zóna umístění obsahu je 1.25m až 5m od uživatele.



Obrázek 11: Optimální vzdálenost umístění obsahu pro zařízení Microsoft Hololens. [8]

Samotné nastavení těchto omezení probíhá přímo v komponentě Kamery editoru Unity. "near cliping"atribut kamery je vhodné nastavit na doporučenou hodnotu 85cm (.85).

[1] https://github.com/Microsoft/MixedRealityToolkit-Unity
[2] https://docs.microsoft.com/en-us/windows/mixed-reality/camera-in-unity

4.4.2 Spatial Awareness

Set komponentů pro nasazení porozumění a mapování okolí (komponenty **Spatial Mapping Renderer a Collider**). **Spatial Mapping Renderer** obsahuje vizualizaci vygenerované mřížky a **Collider** povolí všem objektům reagovat právě na okolní prostředí / dostat se do kolize s reálným světem. Objekty tak dokážou narážet, simulovat fyzikální vlastnosti v závislosti na zmapovaném okolí. Tyto komponenty na sobě nejsou závislé, proto je možné implementovat do aplikace jednotlivě dle potřeby, případně je konfigurovat jednoduše pomocí UI v editoru Unity. [8] [1]

```
SurfaceObserver surfaceObserver;
 void Start () {
     surfaceObserver = new SurfaceObserver();
     surfaceObserver.SetVolumeAsAxisAlignedBox(Vector3.zero, new Vector3(3, 3,
         3));
}
private void OnSurfaceChanged(SurfaceId surfaceId, SurfaceChange changeType,
    Bounds bounds, System.DateTime updateTime)
   {
       switch (changeType)
       {
           case SurfaceChange.Added:
           case SurfaceChange.Updated:
           case SurfaceChange.Removed:
       }
   }
```

Výpis 1: Vytvoření nového surface observer.

4.4.3 Speech Commands

Další důležitou komponentou pro interakci s AR aplikací je rozpoznávání hlasových příkazů uživatele. API Windows Holographic pro Unity nabízí komponenty **KeywordRecognizer** a **PhraseRecognizer** , které obstarávají jednoduchou implementaci ovládání pomocí hlasových příkazů a frází (klíčových slov). Další důležitou komponentou je také **DictationRecognizer** pro obsluhu speech-to-text, tedy přepis hlasu do textu. Lze definovat větší počet instanci těchto tříd tak, aby mohly ve stejnou dobu poslouchat stejné či rozdílné hlasové příkazy nebo přepisovat záznam hlasu. Obsluha těchto komponent je překvapivě velmi jednoduchá a skládá se pouze z vytvoření nové instance s dodáním příkazů k rozpoznaní ve formě textového řetězce, případně

[1] https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping

více příkazů v poli řetězců. Připojení námi definovaných metod k událostem a poté samotné spuštění naší instance[8].[1]

```
[SerializeField]
private string[] m_Keywords;
private KeywordRecognizer m_Recognizer;
...
m_Recognizer = new KeywordRecognizer(m_Keywords);
m_Recognizer.OnPhraseRecognized += OnPhraseRecognized;
m_Recognizer.Start();
```

Výpis 2: Definování nové instance rozpoznávání slov a frází.

4.4.4 Pointer a Cursor

Cursor [2] (dříve GazeCursor) je velmi užitečná komponenta, kterou je vhodné implementovat pro lepší přehled při ovládání UI aplikace či manipulaci s objekty. Jedná se o kurzor, který se vyskytuje uprostřed zorného pole uživatele a dokáže označit objekty, na které uživatel míří **pohledem (gazingem)** [3] předtím, než s nimi začne manipulovat hlasovým či ručním gestem. Jedná se o předem definovaný skript, který je součástí toolkitu, který lze přiřadit objektu, který chceme využívat jako kurzor. [8]

Poté, co je objekt definován jako kurzor, můžeme jednotlivým objektům ve scéně nastavit akci při kolizi s kurzorem - objekt je tedy označen uživatelem. Vhodným příkladem je například transformace z jednoho materiálu do druhého (barvy), aby jsme vizualizovali označení předmětu. **V implementaci si popíšeme jednoduchý transformační skript.**

4.4.5 Billboard a Tagalong [8]

Billboard je objekt, který následuje uživatele z pohledu jeho rotace. Jedná se tedy o prvek, který bude rotovat svou pozici v prostoru podle pozice uživatele a úhlu kamery. Pokaždé když na něj uživatel upře pohled, bude objekt otočen stejnou stranou na něj. [1]

Tagalong Naopak následuje uživatele tak, že je vždy v zorném poli. Tyto objekty ve světě aplikací v rozšířené realitě většinou reprezentují nějaké ovládací prvky, menu nebo nabídku nástrojů (nabídka Start v Hololens je příklad Tagalong objektu). Využití je také vhodné v případně důležitých objektu, které musí zaujmout pozornost uživatele (například notifikace či chybové zprávy). [1]

[1] https://docs.microsoft.com/en-us/windows/mixed-reality/voice-input
[2] https://docs.microsoft.com/en-us/windows/mixed-reality/gestures
[3] https://docs.microsoft.com/en-us/windows/mixed-reality/gaze
[1] https://docs.microsoft.com/en-us/windows/mixed-reality/billboarding-and-tag-along

```
private void Update()
{
Vector3 directionToTarget = Camera.main.transform.position - gameObject.
    transform.position;

switch (PivotAxis)
{
case PivotAxis.X:
 directionToTarget.x = gameObject.transform.position.x;
 break;
case PivotAxis.Y:
 directionToTarget.y = gameObject.transform.position.y;
 break;
}

gameObject.transform.rotation)
gameObject.transform.rotation = Quaternion.LookRotation(-directionToTarget) *
    DefaultRotation;
}
```

Výpis 3: Jednoduchý billboarding objektu v unity

4.4.6 HUD

HUD reprezentuje 2D objekty - převážně obrázkové a textové UI elementy, které budou vždy zobrazeny v zorném poli uživatele a nebudou reagovat na pohyb ani s nimi uživatel nebude nijak pracovat. Jsou umístěny ve vrstvě, která je pevně spjatá s aktuální pozicí kamery a slouží výhradně pro zobrazení nezbytných informací při práci v AR prostředí. V žádném případě by neměly přinášet žádnou obstrukci výhledu, tak jako jsme si popsali v kategorii návrhu AR aplikací.

Tvorba jednoduchého HUD není obtížná, stačí vytvořit nový objekty typu **UI Canvas** - Ten reprezentuje plochu **(plane)** v 3D prostoru na kterou lze umístit 2D elementy. Důležitým nastavení této plochy je její provázání s kamerou, tak aby byla vždy fixně umístěna v zorném poli uživatele. [8]

4.5 Vuforia v Unity

Vuforia představuje naprosto zásadní rozšíření v případě, že požadujeme implementaci rozpoznávání 2D kotevních bodů či trojrozměrných objektů v reálném světě. Dokáže zaznamenat více než jeden cíl a mapovat AR obsah na objekty (například láhve a plechovky, kostky) v reálném světě.

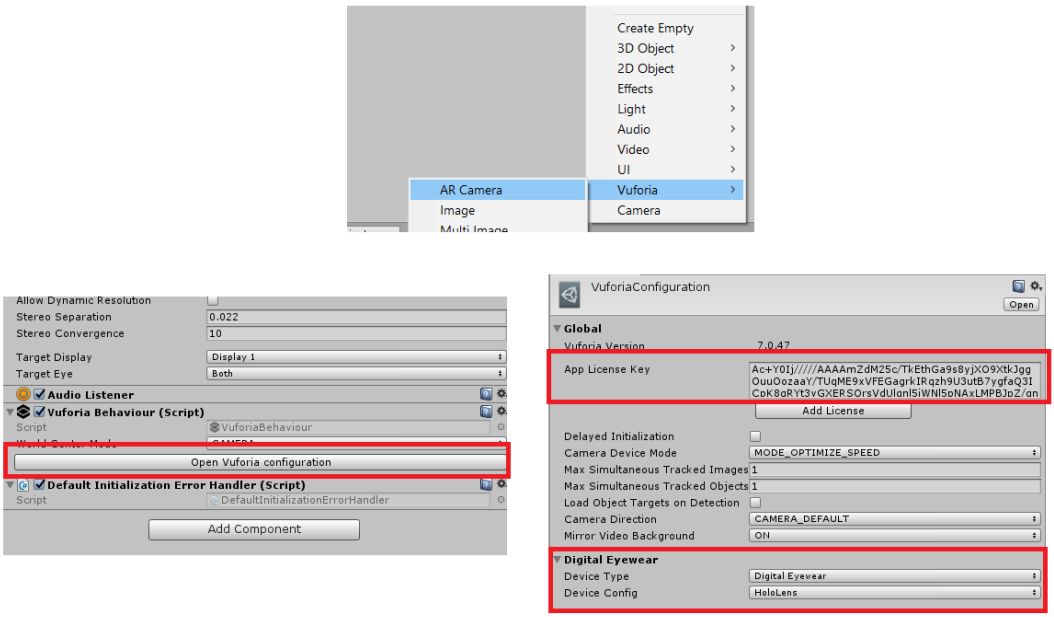
Pomocí tohoto balíku tak můžeme jednoduše detekovat objekty a grafické vzory, které se vyskytují přímo v zorném poli kamery a obohacovat je tak o generovaný AR obsah. [10]

4.5.1 Implementace do projektu

Před samotným použitím je nutné se zaregistrovat na oficiálním vývojářském portále Vuforia a získat API klič, který bude vyplněn v balíku aplikace. Podle vybraného cenového programu, ať už verze zdarma či verze placené, nám budou zpřístupněny funkce v editoru a později i načítány naše obrazové cíle.

Po získání klíče můžeme vložit do naší scény objekt **AR Camera**, který reprezentuje RGB Vstup z kamery. I v tomto případě je třeba tuto kameru označit jako potomka našeho primárního objektu **MixedRealityPlayspace**.

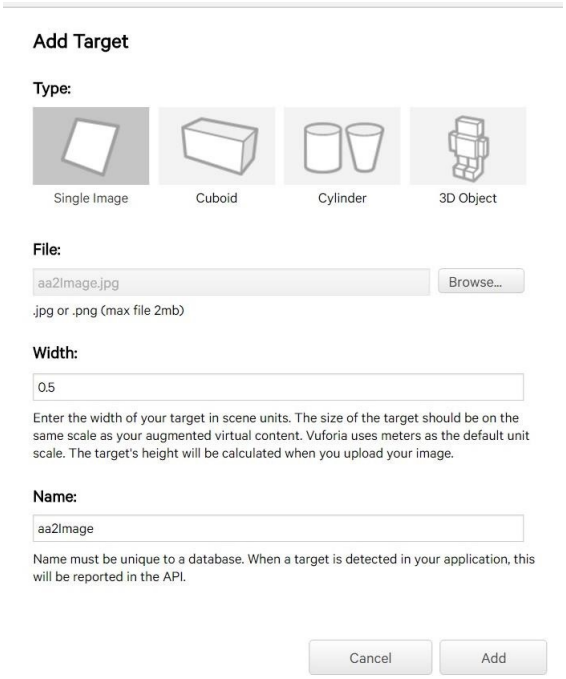
V atributech objektu AR Kamery najdeme samotný přístup do nastavení, kde můžeme specifikovat náš API klíč a určit nastavení pro Microsoft Hololens v záložce "Digital Eyewear". Doporučuje se také přenastavení položky "Camera Device Mode"na optimalizaci pro rychlost, vzhledem k hardwarové limitaci Hololens. [16]



Obrázek 12: Počáteční konfigurace Vuforia v editoru Unity

4.5.2 Nastavení cílových bodů

Cílové body označují objekt v reálném světě (například fotografie či model autíčka) [1], který námi definovaná Vuforia AR camera dokáže rozpoznat ze zdrojové RGB kamery umístěné na Hololens a podle pozice objektu mapovat přiřazený AR obsah do prostoru. Vuforia rozpozná nejen pozici, ale také vzdálenost a rotaci těchto cílových bodů a tyto aspekty dokáže identicky transformovat na náš AR obsah. Definování databáze těchto bodů probíhá opět na portále Vuforia [2] . Takto definované body se pak automaticky zpřístupní v našem editoru, protože jsou svázány s našim API klíčem.



Obrázek 13: Vytvoření nového cílového objektu ve webovém prostředí.

Při definování nového cílového objektu máme na výběr z klasického vstupního obrázku, který lze nahrát ve formátech PNG či JPEG v dostatečném rozlišení (zde patří například QR kódy, fotografie nebo loga). Z krychlového či cylindrického objektu, kde je třeba definovat přesné délky všech stran tohoto objektu (pokud chceme mapovat obsah přímo na reálný trojrozměrný objekt) a v poslední řadě také komplexnějších trojrozměrných objektů, které je třeba naskenovat aplikaci "Vuforia Scanner", která je dostupná zdarma pro mobilní zařízení na platformě Android. Jakmile dokončíme definování našich cílových bodů, je možné přímo ve vývojovém portále pomocí tlačítka “Download All” tuto databázi exportovat a stáhnout ve formátu přijímaném editorem unity (formát .unitypackage). Tento balík pak jednoduše importujeme přímo do našeho projektu.

[1] https://engine.vuforia.com/content/vuforia/en/features.html
[2] https://developer.vuforia.com/

4.5.3 Tag Overlay

"Tag Overlay" nazýváme objekt v našem Unity prostředí, který bude duplikovat pozici vybraného cílového bodu v reálném světě. V momentě kdy AR kamera detekuje v zorném polí cílový bod, bude na pozici bodu zobrazen specifikovaný "Tag Overlay".

Objektu přidáme skript **Image-TargetBehaviour**, poté můžeme přímo vybrat naší databázi bodů a přiřadit specifický cílový bod, který bude reprezentovat náš "TagOverlay" v reálném světě. Takových objektů můžeme samozřejmě definovat více, je ale třeba brát v potaz hardwarové limitace headsetu Hololens. Z testování vychází, že i jeden objekt může velmi výrazně ovlivnit snímkovací frekvenci běžící aplikace.

5 Návrh aplikace

Cílem praktické části diplomové práce je implementace řešení k prezentaci a testování této nové technologické oblasti. **Implementace musí obsahovat jak generování obsahu do scény ve formě vizualizace dat, tak využití hardwarové výbavy zařízení Microsoft Hololens** - snímání okolí a jeho porozumění, implementace anchor-based prvků za pomocí přední HD-RGB kamery a snímání gest uživatele pro dynamickou manipulaci s generovaným obsahem.

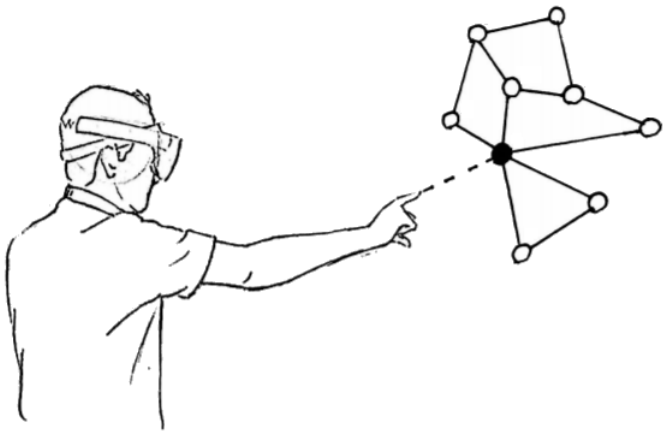
5.1 Problém

Rozdílné vizualizace datových struktur (například grafů a trojrozměrných diagramů) mohou být limitovány při práci na klasickém 2D zobrazovacím zařízení.

5.2 Návrh

Rozšířená realita může do budoucna představovat velmi silného pomocníka datových analytiků při jejich práci s datovými vizualizacemi, kterým dokáže dodat nejen rozměr, ale také zpřístupnit úplně nové techniky práce s daty.

Výsledná aplikace představuje koncept nástroje pro vizualizaci datových struktur v AR. Nabízí možnost jak reprezentace dat 3D bodových grafů, tak síťových datových struktur. S jednotlivými vrcholy grafu lze manipulovat hlasovými příkazy a gesty rukou. Grafy reagují na okolí, včetně simulace elasticity jednotlivých hran a kolize s okolními předměty v reálném světě. Jednotlivé grafické reprezentace lze volně umístit v prostoru, či ukotvit ke předem definovaném QR tagu či identifikátoru. Uživatel tak ve výsledku může manipulovat a zobrazovat data takovým způsobem, jako to na standardních zobrazovacích zařízeních není možné.



Obrázek 14: Nákres konceptu aplikace - síťová data / grafy

5.3 Funkční požadavky

- **Vygenerovat obraz na základě vstupních dat.**

  Aplikace by měla být schopná načíst vstupní data z některého ze standardizovaných formátu pro datasety (ideálně soubory CSV).

- **Různé formy dynamické vizualizace.**

  Aplikace musí obsahovat vizualizaci dvou druhů vstupních dat - Síťová data a trojrozměrný bodový diagram. Předpokládá se, že vizualizace nejsou statické a lze s nimi manipulovat.

- **Implementace rozdílných prvků hardware Hololens.**

  Aplikace musí aktivně využívat funkcionalitu nabízenou zařízením Hololens. Bude implementováno mapování okolí **(spatial mapping a understanding)**. AR obsah by měl reagovat na překrytí okolním prostorem a reálných objektů **(spatial occlusion)** a pomocí frameworku Vuforia implementovat **využití kotevních bodů** jak 2D obrázků, tak pokročilejších vytištěných 3D objektů.

5.4 Microsoft Hololens

Aplikace je implementována za pomocí frameworku **Windows Mixed Reality** na zařízení **Microsoft Hololens** - aktuálně nejpokročilejší hardwarové řešení pro rozšířenou realitu, které pomocí průhledných skel dokáže generované informace prezentovat přímo do zorného pole uživatele. Disponuje hardwarem, který se postará o veškeré výpočty a nevyžaduje připojení k počítači či mobilnímu přístroji. [2] [3]

Obrázek 15: Zařízení Microsoft Hololens

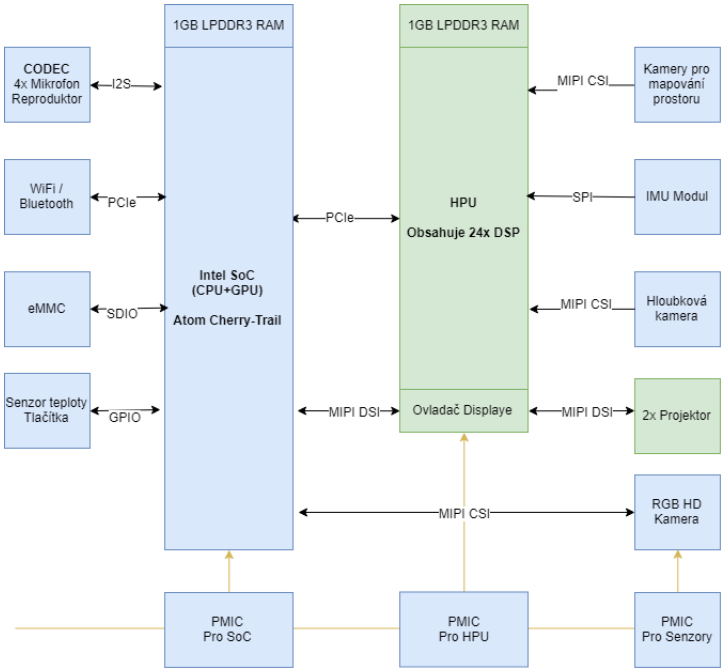
5.4.1 Popis hardware zařízení

Asi nejzásadnější komponentou je **zobrazovací optika**, která se skládá ze dvou průhledných skel sloužící k projekci obrazu pomocí 2 720p 16:9 projektorů (light engines). Tato kombinace nabízí zobrazení až 2.3 milionů světelných bodů s poměrně malým úhlem pohledu **(FOV) přibližně 35 stupňů**, což aktuálně představuje největší limitaci zařízení. Implementace průhledné zobrazovací

jednotky však znamená, že AR obsah je projekčně zobrazen přímo do zorného pole uživatele a není třeba na obsah nahlížet přes display jako je tomu u mobilních zařízení.

Nad průhlednými skly nalezneme **4 kamery specifiky pro mapování prostoru**, společně s 1 hloubkovou ToF kamerou a 1 střední klasickou HD kamerou pro RGB feed. Dalším důležitým prvkem je **IMU** - dedikované zařízení obsahující sety gyroskopu, akcelerometru, magnetometru pro inerciální navigaci, které velmi přesně určuje zrychlení, rychlost a polohu v prostoru. Zařízení dále obsahuje **4 mikrofony** rozmístěné po těle zařízení pro přesný záznam zvuku a **světelný senzor** pro detekci okolního světla pro regulaci jasu projekce.

Hololens je ve své podstatě klasický x86 počítač s mobilním Cherry Trail SoC. Obsahuje low-power **32-bitový x86 procesor** Intel Atom x5-Z8100 s integrovaným GPU (není specifikováno, ale dle ID se jedná o klasický HD Graphics chipset od společnosti Intel.).



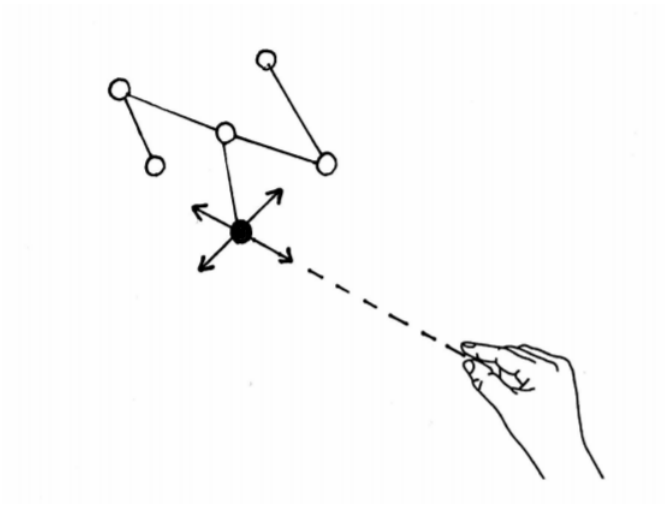
Obrázek 16: Model komponent Microsoft Hololens

Na vysokorychlostní sběrnici PCI-e komunikuje CPU s unikátním koprocesorem zvaným **HPU (Holographic Processing Unit)**, který provádí všechny AR operace a je specifiky navržen pro zpracování enormního množství dat v reálném čase. Obstarává tedy veškeré výpočty a stará se o I/O. Konstantně zpracovává data s vysokou snímkovací frekvencí z IMU a obrazu všech kamer. Obstarává také rozpoznávání řeči a ručních gest uživatele a hlavní CPU těmito operacemi není vůbec zatížen. HPU také dodatečně obsahuje hardwarový ovladač dvou zobrazovacích projektorů a pomocí MIPI DSI komunikuje přímo s integrovaným GPU. Architektura HPU je bohužel zcela utajená a další technické specifikace nejsou veřejné. Dále zařízení obsahuje **2GB DDR3 operační paměti (která je sdílená mezi CPU a HPU), 64GB úložného prostoru a připojením Bluetooth 4.0 a WiFi.**[18]

5.5 Prvky rozšířené reality v aplikaci

5.5.1 Ruční gesta a hlasové příkazy

Nabízí se nám 3 základní modely interakcí s uživatelem. První je rozpoznávání gest rukou pomocí kamer nad průhledovým sklem: tímto lze simulovat klikání na ovládací prvky uživatelského rozhraní či otevíraní kontextových nabídek. Gesta jsou aktuálně dostupná 3 - **"Air tap"** emuluje v aplikaci levé kliknutí myši, **"Bloom"** kdy rozevřením dlaně vyvoláme globální systémovou nabídku Start a lze aplikaci ukončit či minimalizovat, poslední gesto **"Press and hold"** bude ve výsledné aplikace využito pro manipulaci s jednotlivými body v síti, nebo pro přesun grafů ve scéně.



Obrázek 17: Manipulace s vrcholy grafu pomocí gesta "Pinch"

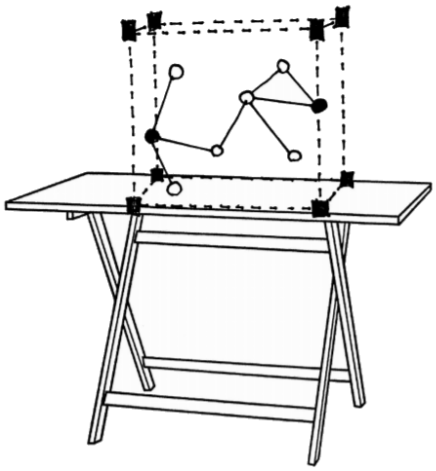
Druhá forma interakce je zvaná **"Gaze"**, kdy uživatel ovládá středový kurzor pomocí pohybu hlavy. Pohledem tedy označujeme samotné prvky vizualizace a ručními gesty s prvky poté manipulujeme. Gazingem v aplikaci lze při kombinaci s gesty také rychle přesouvat jednotlivé vizualizace v prostoru uživatele.

V poslední řadě také využijeme spolehlivého rozpoznávání hlasových příkazů pomoci 4 směrových mikrofonu rozmístěných po těle zařízení. Aby nedošlo ke zbytečné implementaci UI prvků, budou globální příkazy v aplikaci vyvolávány právě pomocí hlasových gest v angličtině.

5.5.2 Snímání tvaru scény a spatial occlusion

Spatial Mapping je jednou z nejdůležitějších součástí produktu Hololens. Díky kolekci senzorů, kterými Hololens disponuje můžeme s poměrně dobrou přesností rozpoznat okolí a objekty v prostředí ve kterém se uživatel aktuálně pohybuje. Náš AR obsah tak díky této funkcionalitě má povědomí o tom, kde je možné se pohybovat a dokáže reagovat na vlivy reálných objektu. Specificky využijeme právě limitování pohybu AR obsahu pouze v reálném prostoru (nebude

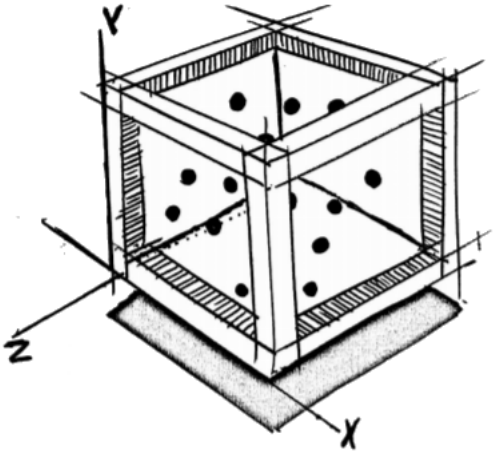
zasahovat do zdí, případně nebude ho možné přesunout na nepřístupné místo). Další zajímavou funkcí je také **Spatial Occlusion**, pomocí které půjde náš AR obsah "zakrýt"pomocí reálných objektů.



Obrázek 18: Jednotlivé grafy reagují na plochy v prostoru a překrytí reálnými objekty.

5.5.3 Detekce základních a pokročilých kotevních bodů

V aplikaci budeme experimentovat s implementaci kotevních bodů na základě rozpoznávání jak jednoduchých 2D objektů (QR Kody, Obrázek) ale také komplexnějších modelů na základě vytrénovaného setu ze vstupního 3D modelu reálného objektu. Zde se bude využívat kombinace jak dat z hloubkové kamery tak RGB kamery. Rozpoznávání bude implementováno za pomocí funkcionality nabízené balíkem Vuforia.



Obrázek 19: Bodový graf umístěný ve 3D tištěném kotevním bodu - prázdné krychli.

6 Implementace aplikace

Základ aplikace musí umět **načíst vstupní data** a **vytvořit atraktivní grafickou reprezentaci** dat do reálného prostředí. Pomocí AR funkcionality by pak vizualizace dat měla **reagovat na podněty prostředí, ruční gesta uživatele** a měla by být **mapovatelná na pozici 2D kotevních bodů a reálných 3D objektů**.

Implementace se skládá z několika Unity scén, které obsahují rozdílné vizualizace dat:

1. **Bodový diagram**

   AR Funkcionalita: Diagram by měl být umístitelný na dvojrozměrné kotevní body a mapovatelný na 3D objekty.

2. **Graf (Síťová data)**

   AR Funkcionalita: S grafem by mělo být možno manipulovat za pomocí ručních gest a měl by reagovat na tvar a reálné objekty okolního prostředí. Umístění do prostoru by mělo být zajištěno pomocí QR kotevního bodu.

6.1 Bodový diagram

6.1.1 Příprava scény a objektů

Do scény umístíme objekt **"MixedRealityPlayspace"**, [27] který obsahuje kameru a předem vytvořenou konfiguraci pro zařízení Microsoft Hololens. Nastavíme průhlednost a pozadí naší scény na černou barvu, tak aby kromě objektů ve scéně nebylo nic zobrazováno na zařízení.

Jednotlivé body diagramu budou reprezentovány objektem **sphere (koule)**, který lze přidat do naší scény pomocí menu **(GameObject -> 3D Object -> Sphere)**. Z tohoto objektu vytvoříme "Prefab"**(Create -> Prefab)**, který pojmenujeme **"ScatterPoint"** - bude reprezentovat bod grafu a můžeme mu definovat specifické vlastnosti. Velikost X,Y,Z našeho budu je nastavena na 0.25 pro přiměřenou velikost.

Posledním objektem scény bude prázdný objekt **"GraphRenderer"**, kterému budou přiřazeny později skripty pro načítání dat a renderování obsahu do scény. Bude reprezentovat nadřazený objekt do kterého bude samotný graf vykreslován.

6.1.2 Načtení vstupních dat

Nativně engine Unity nenabízí žádnou možnost načítání CSV dat, je tedy nutnost si vytvořit vlastní implementaci, která načtení dat zajistí - (**CSVReader.cs**) [21]. Skript bude specifikovaný CSV soubor načítat do datové struktůry slovníku (*List<Dictionary<string, object»*).

```
var list = new List<Dictionary<string, object>>();
TextAsset data = Resources.Load (vstupni_soubor) as TextAsset;
```

Výpis 4: Datová struktura pro uložení vstupních CSV dat a načtení vstupu.

Skript začíná detekcí počtů řádků a rozdělení jednotlivých řádku CSV souboru do pole. Následně dochází k rozdělení hlavičky CSV souboru a tak vydefinování počtu sloupců a jejich názvu. Pro každý sloupec bude ve výsledné datové struktuře vytvořen objekt, který bude obsahovat patřičné hodnoty sloupce ze vstupního souboru.

```
var lines = Regex.Split (data.text, LINE_SPLIT_RE);
if(lines.Length <= 1) return list;
var header = Regex.Split(lines[0], SPLIT_RE);
```

Výpis 5: Rozdělení řádků souboru a rozdělení hlavičky souboru.

Pro každý řádek ve vstupním souboru je proveden v cyklu proces zpracování dat. Pomocí regulárního výrazu je zpracovávaný řádek rozdělen (na základě specifikovaného oddělovače) do pole reprezentující jednotlivé hodnoty v řádku. Jednotlivé hodnoty jsou poté ve výstupní struktuře přiřazeny ke svému sloupci.

```
for(var i=1; i < lines.Length; i++) {
        var values = Regex.Split(lines[i], SPLIT_RE);
        if(values.Length == 0 ||values[0] == "") continue;
        var entry = new Dictionary<string, object>();
        for(var j=0; j < header.Length && j < values.Length; j++ ) {
           string value = values[j];
           value = value.TrimStart(TRIM_CHARS).TrimEnd(TRIM_CHARS).Replace("\\"
               , "");
           object finalvalue = value;
           int n;
           float f;
           if(int.TryParse(value, out n)) {
              finalvalue = n;
           } else if (float.TryParse(value, out f)) {
              finalvalue = f;
           }
           entry[header[j]] = finalvalue;
        }
        list.Add (entry);
     }
```

Výpis 6: Zpracování řádků CSV souboru

6.1.3 Příprava dat

Prvním krokem je samozřejmě využití našeho CSV skriptu, který jsme si definovali v předchozím kroku a který nám data načte do datové struktury s kterou můžeme nadále pracovat. Vstupním parametrem pro náš skript pro načítání souboru je cesta k souboru.

Definicí veřejných (public) proměnných ve skriptu můžeme tyto proměnné modifikovat přímo v UI editoru Unity, toto zajišťuje velkou flexibilitu při ladění aplikace a budeme tak připravovat každý skript v aplikaci.

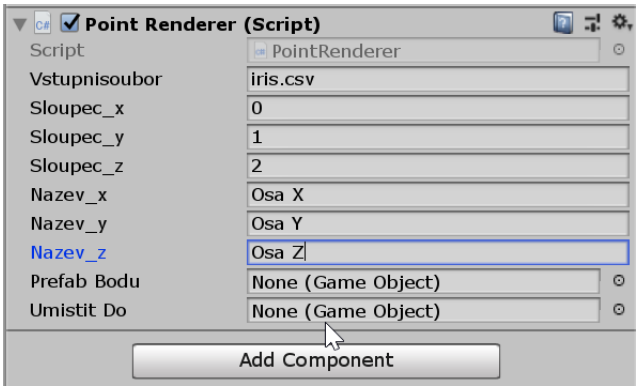
Při načítáni našich dat začneme definicí názvu X, Y, Z os našeho grafu. Stejně tak si definujeme číselné proměnné které budou označovat číslo sloupce v CSV souboru, který bude přiřazen dané ose grafu. Všechny tyto proměnné jsou veřejné a můžeme s těmito atributy skriptu pracovat přímo v editoru Unity (Obrázek 20) [20].

```
public class DataPlotter : MonoBehaviour {
 public string input;
 private List<Dictionary<string, object>> data;
 public int Col_x = 0;
 public int Col_y = 1;
 public int Col_z = 2;
 public string Name_x;
 public string Name_y;
 public string Name_z;

 void Start () {
 data = CSVReader.Read(input);
 List<string> col_names = new List<string>(data[1].Keys);
 Name_x = col_names[Col_x];
 Name_y = col_names[Col_y];
 Name_z = col_names[Col_z];
 }
}
```

Výpis 7: Základní struktura skriptu pro renderování bodů grafu.

Když jsou naše vstupní data načteny ve zpracovatelné podobě a máme připraveny i jednotlivé reprezentace os našeho grafu můžeme se pustit do implementace umístění jednotlivých bodů grafu do AR scény. Definicí veřejné proměnné typu GameObject můžeme zpřístupnit přiřazení prvku typu GameObjectu (prefab) přímo v prostředí editoru Unity. Následně tak můžeme našemu skriptu přiřadit objekt, který bude ve scéně reprezentovat bod v našem grafu (ScatterPoint).



Obrázek 20: Veřejné proměnné našeho skriptu modifikovatelné v prostředí editoru Unity

Pro vytvoření nového objektu scény využíváme Unity příkaz **"Instantiate"**. V této funkci definujeme o jaký objekt se jedná, jaké jsou jeho X,Y,Z koordináty (souřadnice) ve scéně a jaká je jeho rotace.

```
Instantiate(ScatterPoint, new Vector3(x, y, z), Quaternion.identity);
```

Výpis 8: Umístění bodu grafu (kopie prefab) do scény.

Koordináty jsou specifikovány pomocí struktury **Vector3**, která se využívá v enginu Unity pro definici trojrozměrného vektoru - pozice, případě určení směru ve scéně. Definovat ji lze pomocí konstruktoru *new Vector3(X,Y,Z)*. Dodatečně lze taky využít jeho vlastností *(back, down, forward, left atd.)* pro rychlou manipulaci s daným vektorem. Jakákoliv rotace objektu je reprezentována v Unity pomocí **"Quarterion"**. Jednoduše tak lze manipulovat také s rotaci daného objektu pomocí vlastností *"LookRotation", "Angle", "Euler", "Slerp"* a další.

Při zacyklení tohoto procesu a vytvoření objektu pro každý řádek našeho vstupního souboru a přiřazení souřadnic každého bodu na základě hodnot jednotlivých sloupců v řádku, jsme vytvořili jednoduchou grafickou reprezentaci všech bodů ze vstupního souboru do naší scény.

```
for (var i = 0; i < ptList.Count; i++)
{
float x = System.Convert.ToSingle(ptList[i][Name_x]);
float y = System.Convert.ToSingle(ptList[i][Name_y]);
float z = System.Convert.ToSingle(ptList[i][Name_z]);
Instantiate(ScatterPoint, new Vector3(x, y, z), Quaternion.identity);
}
```

Výpis 9: Vykreslení bodu CSV do scény.

Implementaci lze dále optimalizovat úpravou vytváření bodů tak, aby jednotlivé body byly rozdílné instance specifikovaného prefabu. Nejen, že se jedná o efektivnější implementaci, ale dovolí nám to vy výsledku manipulovat flexibilně i s jednotlivými body grafu ve scéně.

```
GameObject dataPoint = Instantiate(ScatterPoint, new Vector3(x, y, z),
    Quaternion.identity);
```

Výpis 10: Umístění bodu grafu do scény.

6.1.4 Normalizace dat

Momentálně jsou naše body umístěny ve scéně na základě surových vstupních dat a je nutností naše body nějak normalizovat tak, aby jejich pozice v grafu nebyla vázána k celé scéně, ale pouze k velikosti grafu, jakožto nadřazeného objektu ve scéně.

V našem skriptu **DataPlotter** začneme tím, že si určíme minimum a maximum na každé ose na základě načtených dat. Vytvoříme si tedy funkce **FindMinVal** a **FindMaxVal**, které pomocí cyklu postupně porovnávají všechny hodnoty každé osy a uloží do výstupní proměnné maximální či minimální hodnotu.

V momentě kdy máme určeno minimum a maximum každé osy našeho grafu můžeme jednoduše pomocí následujícího vzorce vypočítat normalizovanou pozici našeho bodu v grafu:

$$pozice(x) = \frac{x - xmin}{xmax - xmin}$$

```
private float FindMaxVal(string col)
{
    float maxV = Convert.ToSingle(ptList[0][col]);
    for (var i = 0; i < ptList.Count; i++)
    {
        if (maxV < Convert.ToSingle(ptList[i][col]))
            maxV = Convert.ToSingle(ptList[i][col]);
    }
    return maxValue;
}
```

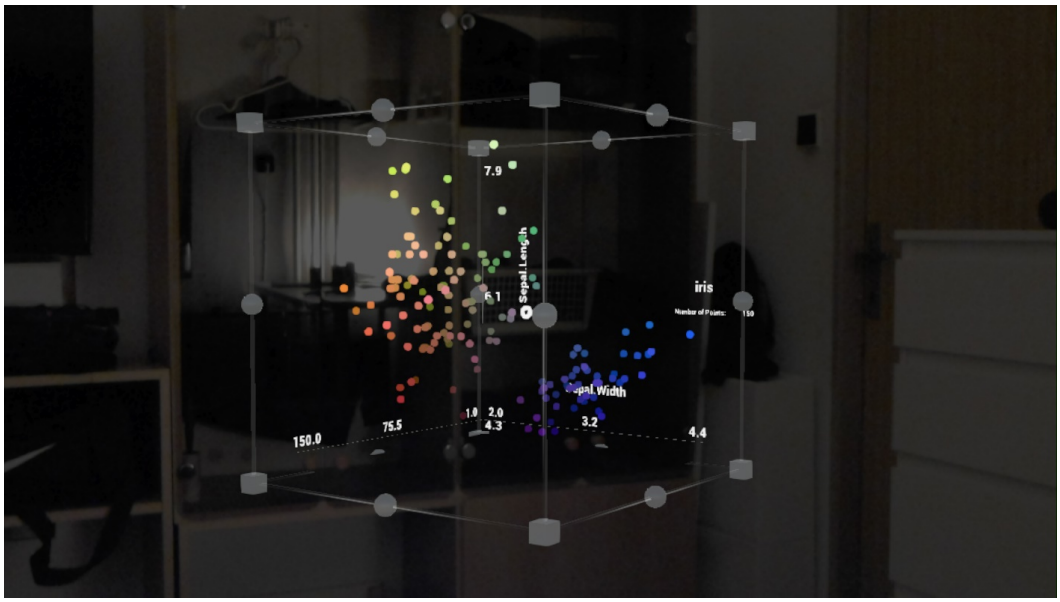
Výpis 11: Nalezení maxima ve specifikovaném sloupci

Výhodou normalizace je také možnost velmi flexibilního škálování velikosti našeho grafu pouze pomocí násobením vektoru (hodnotou v proměnné zoom).

```
public float zoom = 1;
float x = (System.Convert.ToSingle(ptList[i[Name_x]) - xMin) / (xMax - xMin);
float y = (System.Convert.ToSingle(ptList[i[Name_y]) - yMin) / (yMax - yMin);
float z = (System.Convert.ToSingle(ptList[i[Name_z]) - zMin) / (zMax - zMin);

GameObject dataPoint = Instantiate(PointPrefab, new Vector3(x, y, z) * zoom,
    Quaternion.identity);
```

Výpis 12: Umístění bodu grafu na normalizované pozici s možností škálování.



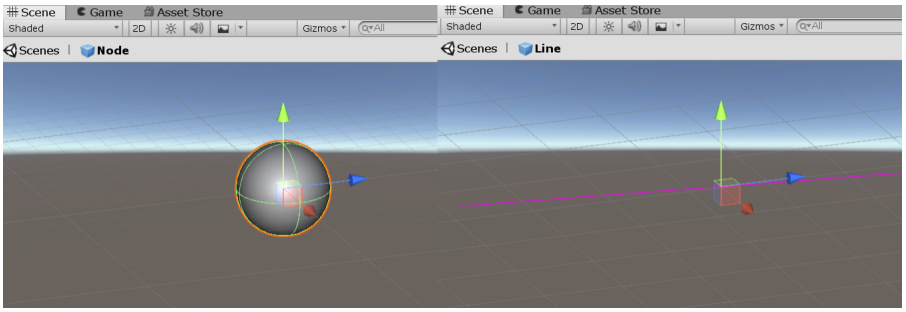
Obrázek 21: Výsledný bodový graf na zařízení Hololens

6.2 Grafy

6.2.1 Příprava scény a objektů

Do scény opět umístíme **"MixedRealityPlayspace"**,[27] který obsahuje kameru a předem vytvořenou konfiguraci pro zařízení Microsoft Hololens. Přidáme objekt zvaný **"GraphOverlay"**, který za běhu scény bude obsahovat vygenerovaný graf. Jednotlivé vrcholy grafu a jejich hrany jsou poté vygenerovány v reálném čase jako jednotlivé objekty scény, což umožňuje dynamickou reprezentaci a možnost manipulace s grafem ve scéně při běhu vizualizace.

Jednotlivé vrcholy grafu jsou reprezentovány objektem zvaným **"Node"**, stejně jako u předchozí vizualizace se jedná o standardní objekt typu "Sphere". Hrany mezi těmito vrcholy jsou vykreslovány pomocí komponenty **"Line Renderer" (GameObject -> Create Empty)** a přiřazení komponenty **(Line Renderer)**. Tato komponenta generuje přímou čáru mezi dvěma specifikovanými objekty ve scéně. Čára se aktualizuje za běhu aplikace při každém snímku, takže pokud dojde k přesunu bodu za běhu aplikace, přizpůsobí se tomu i hrany grafu.



Obrázek 22: "Node"a "Line"ze kterých je ve scéně složen graf.

6.2.2 Příprava vstupních dat

Pro načtení dat z CSV souboru opět využijeme náš skript **CSVReader.cs**, který jsme si implementovali pro předchozí vizualizaci.

Pro uchovávání jednotlivých vrcholů a hran grafu je namísto jednoho standardního slovníku v tomto případě implementována datová strukturu zvanou **"Adjecency List"** [23].

Slovník zvaný **vertexDict** *(Dictionary<int, List<int»();)* uchovává jednotlivé vrcholy grafu. Obsahuje klíč (key) tohoto vrcholu a k němu přiřazené hodnoty (values) ve formě listu klíčů jeho sousedících vrcholů, mezi kterými se vyskytuje hrana.

```
private List<List<int>> _vertexList = new List<List<int>>();
private Dictionary<int, List<int>> _vertexDict = new Dictionary<int, List<
    int>>();
```

Výpis 13: Ukládání jednotlivých vrcholů grafu a jejich hran.

Implementace přidání nové hrany mezi vrcholy nejprve ověří, zda specifikované identifikátory vrcholů jsou správné a označují vrcholy, které již v našem listu existují. Samozřejmě rozdělujeme počáteční i cílový vrchol v případě, že budeme našim adjecency listem chtít reprezentovat orientovaný graf.

```
List<int> startVertex = _vertexDict.ContainsKey(startKey) ? _vertexDict[
    startKey] : null;
List<int> endVertex = _vertexDict.ContainsKey(endKey) ? _vertexDict[endKey] :
    null;
      startVertex.Add(endKey);
      endVertex.Add(startKey);
```

Výpis 14: Přidání nové hrany mezi vrcholy

Kromě přidávání nových vrcholů a hran implementace také obsahuje odebrání jednotlivých vrcholů **(RemoveVertex)**, odebrání samostatné hrany **(RemoveEdge)**, ověření zda mezi dvěma vrcholy existuje hrana **(isAdjacent)**, získání seznamu všech hran pro daný vrchol **(GetEdgesForVertex)** a získání stupně vrcholů**(GetVertexDegree)**. Pomocí tohoto setu funkcí lze vizualizovat téměř každý algoritmus a operaci prováděnou na grafech.

```
public void RemoveVertex(int key)
   {
      List<int> vertex = _vertexDict[key];
      int vertexNumAdjacent = vertex.Count;
      for (int i = 0; i < vertexNumAdjacent; i++)
      {
         int neighbourVertexKey = vertex[i];
         RemoveEdge(key, neighbourVertexKey);
      }
      _vertexList.Remove(vertex);
      _vertexDict.Remove(key);
   }
```

Výpis 15: Ukázka odebrání vrcholu a všech jeho hran

6.2.3 Generování vizualizace

Kromě načtení existujícího grafu ze souboru je implementováno také náhodné generování grafu pro rychlou demonstraci na základě několika specifikovaných atributů. Tuto funkcionalitu zajišťují funkce **GenerateGraph()** a **RandomPlaceNodes()**. V generátoru lze specifikovat počet vrcholů a šanci vytvoření hrany mezi nimi. Pomocí funkce **RandomPlaceNodes** lze vyvolat proces náhodného umístění specifikovaného počtů vrcholů do scény a poté vyvolat komponentu "Line Renderer"pro vrcholy, která ve scéně vykreslí jednotlivé hrany mezi vrcholy.

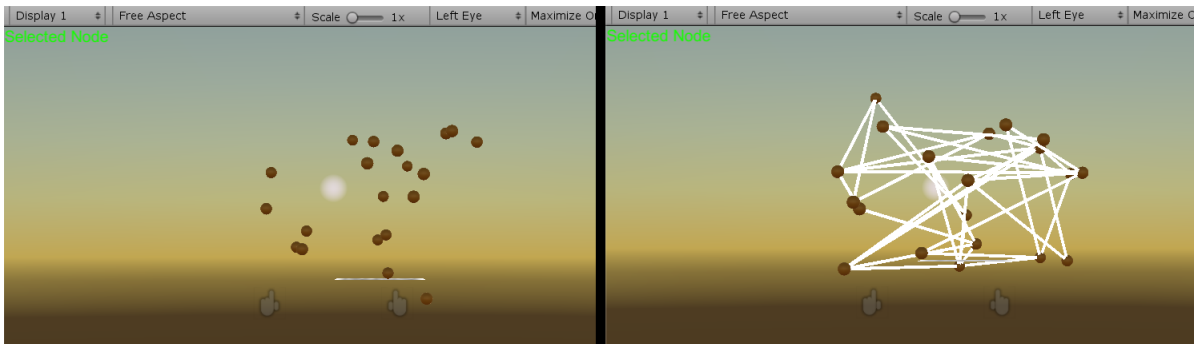
Samostatné umístění jednotlivých vrcholů do scény je velmi podobné implementaci v našem bodovém diagramu v předchozí kapitole.

```
void randomlyPlaceNodes(){
int numNodes = masterNodeList.Length;

for (int i = 0; i < numNodes; i++) {
if (i != 0) { adjacencyList.AddVertex (i);}

GameObject myNodeInstance = Instantiate (Resources.Load ("Node"),new Vector3 (
    Random.Range (-SPREAD, SPREAD), Random.Range (-SPREAD, SPREAD) + DIST,
    Random.Range (-SPREAD, SPREAD)),Quaternion.identity) as GameObject;
masterNodeList [i] = new Node (myNodeInstance, i);
myNodeInstance.transform.parent = GameObject.Find("GraphOverlay").transform;
       }
   }
```

Výpis 16: Umístění náhodných vrcholů grafu do scény



Obrázek 23: Základní vykreslení samostatných vrcholů grafu ve scéně a vykreslení včetně line-rendereru reprezentující hrany.

6.2.4 Manipulace s grafem

S jednotlivými vrcholy grafu je možnost pohybovat za běhu aplikace. Pomocí gest rukou lze jednotlivé vrcholy grafu uchopit a přesunout v AR prostředí. Na tuto akci samozřejmě reaguje také komponenta Line Renderer, která nové pozici vrcholu musí přizpůsobit i jeho hrany při každé aktualizaci snímku. Funkcionalitu lze jednoduše implementovat za pomocí komponenty **"Manipulation Handler"** [27] ze setu MRTK. Při detekci vybraného gesta rukou (v našem případě "Tap and Hold") dovolí manipulovat s pozicí vybraného objektu ve scéně.[24]

Obrázek 24: Komponenta "ManipulationHandler"přiřazená k našemu vrcholu v grafu.

Aby manipulace s vrcholy nebyla nepřehledná vytvoříme skript, který provede změnu materiálu (barvy) vrcholu, na který právě uživatel míří kurzorem. Vytvoříme si nový skript **"GazeSelection"**, který bude potomkem třídy **"GazeTransitionBase"** [27] ze setu MRTK, která zajišťuje, že bude proveden vždy, kdy dojde ke kolizi kurzoru a objektu kterému je skript přiřazen.

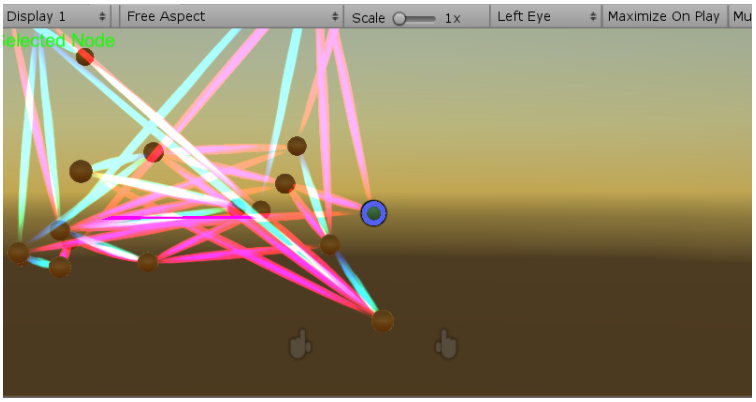
Ve skriptu si vydefinujeme dvě veřejné proměnné typu **"Material"**, které budou označovat materiály mezi kterými se provede přechod v případě označení objektu gaze kurzorem. Samotná změna materiálu objektu je provedena pomocí funkce **Material.Lerp**, která interpoluje mezi dvěma materiály po dobu volání funkce **Update()** v našem skriptu ve specifikovaném časovém intervalu (transitionFactor).

```
public class GazeSelection : GazeTransitionBase {
public Material activeMaterial;
public Material inactiveMaterial;
float transitionFactor = 2.0f;
private Renderer Renderer { get; set; }

void Start () {
Renderer = GetComponentInChildren<Renderer>();
   }

void Update () {
if (Renderer == null)
return;
Renderer.material.Lerp(inactiveMaterial, activeMaterial, transitionFactor);
}
}
```

Výpis 17: Jednoduchá změna materiálu označeného objektu scény



Obrázek 25: Vrcholy transformují svou barvu po označení gaze kurzorem. Při aplikaci ručního gesta lze vrchol ve scéně přesunout.

6.2.5 Interakce s reálným prostředím

Klíčová komponentu kterou využijeme pro implementaci interakce s reálným prostředím je **"Spatial Awareness System"**, [25] [27] která zajišťuje detekci ploch a nastavení hranic reálného světa tak, aby generovaný obsah mohl reagovat na podněty okolního prostředí. K Spatial Awarness Systemu se přímo váže registrace takzvaného **"Observeru"** který vytváří aproximační model (mesh) okolí jakožto nový objekt naší Unity scény, který dokáže přímo v aplikaci reprezentovat reálný svět.

*Volitelně lze povolit také komponentu **"Boundary Visualization"**, která dokáže provést jednoduchou grafickou vizualizaci mapovaného prostředí přímo do aplikace. Dokáže jednotlivé plochy, dostupné okolí, stěny a stropy vizualizovat pomocí zvoleného materiálu v enginu Unity.*

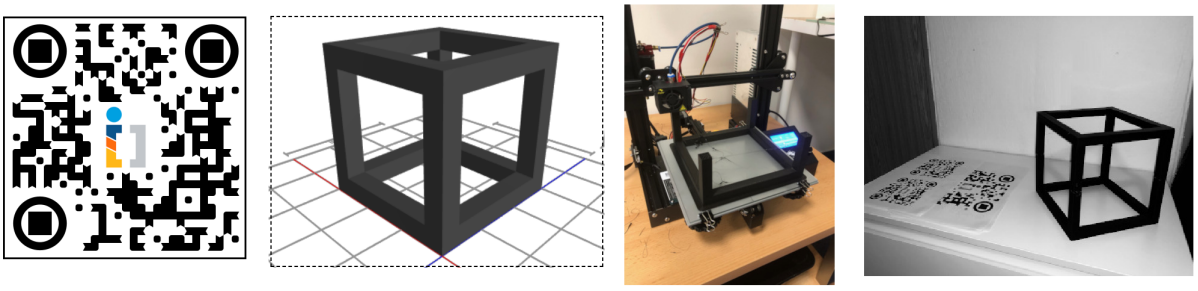
Pro objekt naší scény po kterém vyžadujeme kolizi s reálným prostředím je potřeba přiřadit komponentu **"Spatial Mapping Collider"**.

6.3 Vuforia - Kotevní 2D body a 3D objekty

V momentě kdy se v našem objektu **"GraphRenderer"** zobrazuje graf, můžeme se pustit do integrace kotevních bodů. Prvním krokem je přidání objektu ARcamera **(GameObject>Vuforia> AR Camera)**, tento objekt reprezentuje vstupní RGB feed z kamery ve kterém bude prováděno rozpoznávání objektů. V nastavení našeho projektu povolíme podporu pro balík Vuforia (Vuforia Augumented reality Supported) a v nastavení ARkamery vybereme konfiguraci pro Microsoft Hololens (Device Type and Config). **Další konfiguraci balíku Vuforia provedeme podle popisu z kapitoly 4.7 této diplomové práce.**

Integrace do Unity projektu je poměrně jednoduchá, druhou části je pak konfigurace samotných kotevních bodů. Pro náš bodový diagram si vydefinujeme dva druhy kotevních bodů - jeden standardní 2D a druhý pro pokročilé mapování obsahu do reálných trojrozměrných objektů. [22]

- **2D Bod** - QR kód na papíře. Graf se zobrazí tam kde je detekován QR kód ve scéně.
- **3D Objekt** - Model kostry kvádru z CADu vyrobené na 3D tiskárně. Graf se bude zobrazovat přímo v detekovaném reálném objektu reprezentující okraje grafu.



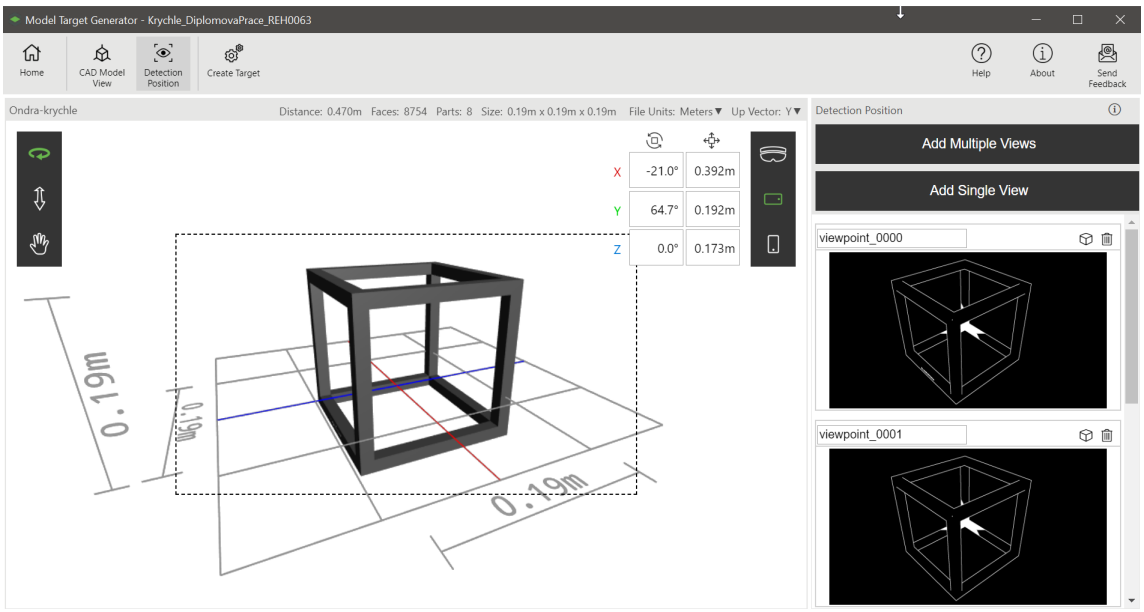
Obrázek 26: QR kotevní kód a 3D tištěný trojrozměrný kotevní objekt

QR kód můžeme pomocí webového nástroje na vývojářském portálu Vuforia [1] nechat zpracovat a převést do balíku importovatelného do našeho Unity projektu (proces je popsán blíže v kapitole 4.7.2 této diplomové práce.)

Vytvoření trojrozměrného kotevního bodu/objektu je o komplikovanější a vyžaduje stažení nástroje **Vuforia Model Target Generator** . [2]

[1] https://developer.vuforia.com/
[2] https://developer.vuforia.com/downloads/tool

6.3.1 Model Target Generator

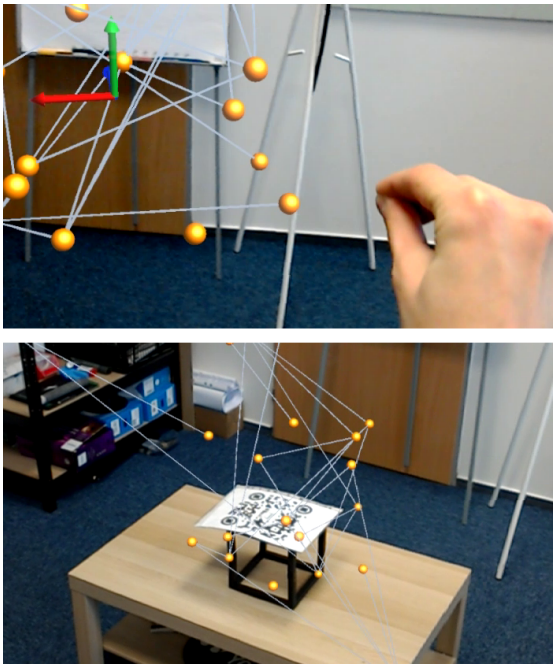


Obrázek 27: Prostředí Vuforia Model Target Generator s modelem našeho kotevního objektu a nastavením jeho detekčních úhlů.

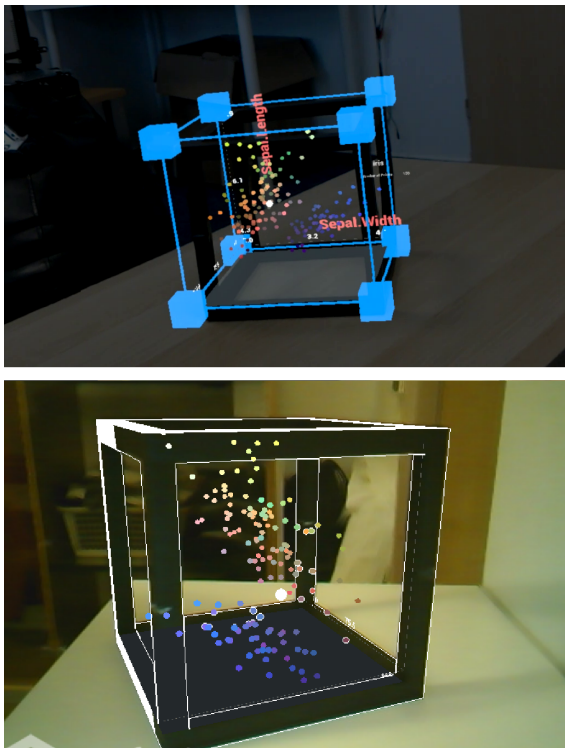
Model Target Generator je nástroj pro balík Vuforia, který dokáže na základě vstupního 3D modelu trénovat dataset pro rozpoznávání tvaru. V tomto nástroji můžeme načíst vstupní model objektu v jakémkoliv ze standardních CAD formátu, specifikovat detekční úhly a určit rozměry reálného objektu. Výsledkem je opět vygenerování balíku importovatelného do našeho Unity projektu. [28]

Importované balíky (jak z 2D bodu tak 3D objektu) poté přiřadíme k nově vytvořeným objektům scény **"Image Target"** a **"Object Target"**. Tyto objekty představují implementaci rozpoznávání obrázků a objektů v pohledu ARcamery. Jejich pozice ve scéně bude mapována 1:1 k pozici detekovaného objektu v reálném prostoru, respektive pozic objektu v RGB feedu kamery našeho zařízení. Přiřazením našeho objektu **"GraphRenderer"** jako potomka jednoho z těchto objektů docílíme umístění našeho grafu do scény na pozici detekovaného objektu, v případě že je objekt detekován.

6.4 Výsledné vizualizace - Microsoft Hololens



Obrázek 28: Výsledná interaktivní vizualizace grafu na Microsoft Hololens



Obrázek 29: Bodový diagram mapovaný na trojrozměrný reálný objekt na Microsoft Hololens

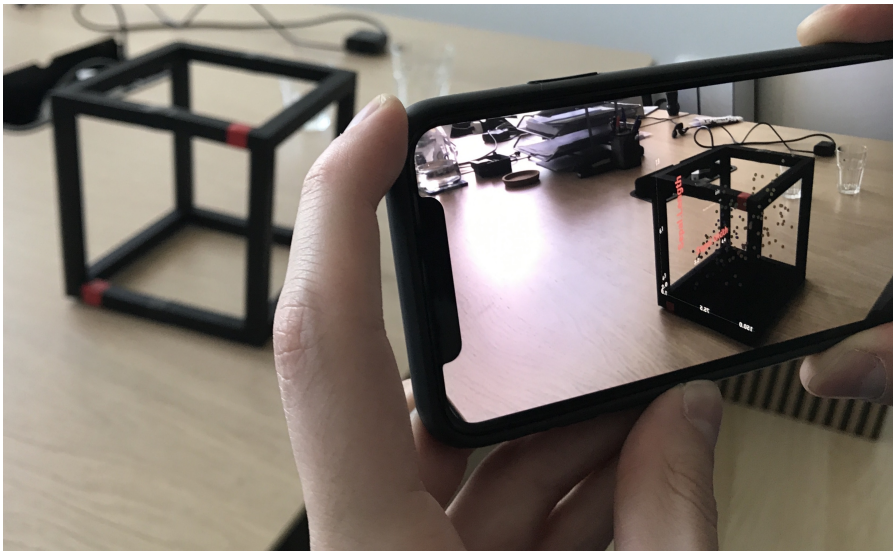
6.5 Port pro Apple ARkit

Přesto, že se zadání diplomové práce věnovalo specificky platformě WMR, tak v rámci praktické části byla výsledná prezentace bodového grafu a jeho mapování na trojrozměrný kotevní bod portována navíc i na zařízení společnosti Apple a její softwarovou AR platformu **ARkit**. Výsledkem je identická aplikace běžící na zařízeních Apple iPhone a iPad. **Výborně reprezentuje výkonnostní rozdíl a benefity, které přináší AR optimalizované procesory A11/A12 Bionic na nových iOS zařízeních.** Slouží jako srovnání obou platforem a rozdílných druhů zařízení.

Díky tomu, že Unity i framework Vuforia je plně kompatibilní s operačními systémy iOS, bylo sestavení aplikace na novou platformu velmi efektivní a v průběhu se nevyskytla žádná velká překážka. V nastavení projektu v editoru Unity musí dojít ke **změně cílové platformy**, kompletní **eliminace MRTK prvků** a **nové nastavení AR Kamery**, které odpovídá cílovým mobilním zařízením. [29]

Nespornou výhodou editoru Unity je přenositelnost implementace. Přesto, že platformy jsou zcela odlišné, došlo pouze k minimálním modifikacím v kódu, který zajišťuje vykreslování grafu do scény. Většina práce při portování odpovídala převážně nastavení projektu a přenosu do vývojového prostředí společnosti Apple - Xcode.

Microsoft Hololens má velkou výhodu ve své zobrazovací jednotce, bohužel značně doplácí na svůj výpočetní výkon a kvalitu vstupní RGB kamery. Za běhu aplikace na iPhone jde výrazně vidět jak velký přínos má výkonný procesor jako A11 Bionic s optimalizací pro AR operace, společně s několika velmi kvalitními kamerami (Specificky na iPhone X/XS). Běh i snímání kotevního bodu je mnohonásobně plynulejší s mnohem kratší reakční dobou. Práce s aplikací je ve výsledku mnohem flexibilnější. Tyto nedostatky na zařízení Hololens by měla avizovat nová hardwarová revize v roce 2019.



Obrázek 30: Identická aplikace běžící na iPhone X (ARkit)

7 Zhodnocení

7.1 Výsledná aplikace

Hlavní myšlenou této diplomové práce je realizace různých forem vizualizace a interakce s důrazem na implementaci prvků rozšířené reality. Za tímto účelem došlo k implementaci aplikace **"HoloGraph"** pomocí herního enginu Unity a frameworku Windows Mixed Reality se specifickým zaměřením na zařízení Microsoft Hololens. Výsledná aplikace demonstruje projekci obsahu do reálného prostředí, manipulaci s obsahem za pomocí ručních gest a příkazů, mapování okolí pomocí Spatial Mappingu a implementaci dvojrozměrných a trojrozměrných kotevních bodů pro umístění obsahu.

7.2 Problémy při vývoji

V průběhu realizace diplomové práce jsem byl svědkem vývoje samotného Mixed Reality Toolkitu (dříve HoloToolkitu) i kompletní integrace do enginu Unity. Vzhledem k tomu, že platforma je stále velmi mladá a programovací standardy pro WMR ještě nebyly pevně stanoveny, byl vývoj aplikace pro Microsoft Hololens poměrně náročný z hlediska kompatibility a opakovaného přepracovávání a implementace nové funkcionality.

Po dobu vývoje aplikace se platforma neustále měnila, nezůstalo ji ani původní jméno, a bylo velmi obtížné se vůbec orientovat ve verzování. **Jednotlivé verze SDK mezi sebou nebyly kompatibilní** a **v průběhu roku 2018 došlo ke kompletnímu přepracování původního "HoloToolkit" na nový "Mixed Reality Toolkit".** Výsledkem bylo, že celá implementace aplikace v diplomové práci musela být přepracována. Dřívější verze Mixed Reality Toolkit implementovaly jednotlivé funkce úplně jinak, případně je vůbec neobsahovaly a bylo třeba vyvinout vlastní rozšíření, které se poté v nových verzích opět staly nepoužitelnými.

Dokumentace a články z různých zdrojů na tento nešvar taktéž trpěly. **Do dnešní doby je velmi obtížné najít příručky, případně kvalitní zdroje, které se věnují aktuální verzi MRTK pro Hololens.**

Samotné zařízení z hlediska operačního systému od roku 2017 doznalo neuvěřitelného pokroku. Původní verze operačního systému s aktuální verzí je téměř nesrovnatelná hlavně z hlediska rychlosti a kompatibility. Bohužel na to trpí velký počet Hololens aplikací v Microsoft Store, které jsou očividně postavené na starších řešeních a jejich fungování je při nejlepším pochybné. **Ve výsledku je tak Microsoft Store plný špatně fungujících nebo nedopracovaných aplikací.**

Dalším zásadním problémem, se kterým jsem se při vývoji setkal byla **velká časová náročnost sestavení aplikace** a nahrání na zařízení pro testování. V mnoha případech trvalo celé přestavení projektu a nasazení na zařízení více než 20 minut, bez ohledu jak výkonný PC byl při vývoji použit. Doprovázeno s původní nestabilitou komponent v balíku MRTK toho činilo vývoj velmi frustrujícím. Užití emulátoru zařízení Hololens je taky silně individuální, protože

jsem se v mnoha případech setkal s tím, že poskytnutý výkon emulátoru neodpovídal reálnému zařízení.

Naštěstí v aktuální verzi editoru Unity lze pomocí doprovodné aplikace **"Holographic Remoting Player"** možné využít **"Remote Play"** přímo v editoru, který dokáže promítat scénu přímo do zařízení Hololens přes Wi-Fi síť, přesto, že je samotná aplikace renderována na PC. Toto je obrovsky užitečným nástrojem který vývoj podstatně ulehčuje při testování.

Z vlastní zkušenosti v žádném případě nedoporučuji spouštění aplikací v "Debug" sestavení na zařízení Hololens. Bohužel při běhu aplikace v debug režimu je díky mnoha komponent užitým pro ladění výkon aplikace natolik ovlivněn (v mnoha případech z hlediska snímkovací frekvence až o -700 procent), že běž aplikace nereprezentuje stabilitu a výkon na zařízení a činí celé ladění aplikace velmi nepohodlným. Doporučuji tak implementaci jiných metod ladění, které tolik netíží výkon samotné aplikace. Tento fakt není dokumentován a představoval při vývoji po velmi dlouhou dobu naprosto zásadní problém, který měl velmi jednoduché, avšak neznámé řešení.

7.3 Aktuální stav ekosystému WMR

Přesto, že se počáteční chaos nejvíce podepsal na problémech s dokumentací a standardizací vývoje, je dnes konečně situace jiná a **WMR ekosystém se koncem roku 2018 značně ustálil**. Je očividné, že společnost Microsoft do svého Mixed Reality Toolkitu vložila obrovské úsilí a balík je v aktuální verzi velmi dobře vyřešen. Obsahuje všechny důležité komponenty pro rychlý vývoj aplikací, dokumentace je konečně přehledná a čistá a bez debaty žádný z konkurentů nenabízí takto rozšířený a otevřený balík nástrojů pro vývoj AR.

7.4 Hardware Hololens

První verze zařízení "Hololens" je bez pochyby určena pro vývojáře, kteří se nebojí experimentovat a než aby šlo o platformu, na které je vhodné vyvíjet finalizovaná řešení, jedná se spíš o zařízení pro testování konceptu a finalizovaní představy samotného ekosystému rozšířené reality společnosti Microsoft. Zařízení je stále omezeno primárně z hlediska výpočetního výkonu, který podstatně omezuje vývoj komplexnějších aplikací a limitovanou zobrazovací jednotkou, která poskytuje velmi úzký úhel viditelnosti. Komfort užívání zařízení po delší době je taky silně ovlivněn větší váhou jednotky a značnou nepohodlností pro uživatele, kteří nosí dioptrické brýle.

S příchodem nového Hololens 2 koncem roku 2019 je tak představa AR pro platformu WMR mnohem jasnější a předpokládám, že finalizací konceptu WMR a novým výkonnějším hardwarem bude vývoj mnohem pohodlnější a konečně se objeví aplikace, které odemknou plný potenciál konceptu rozšířené reality.

7.5 Programatorská přívětivost

Vývoj aplikací pro rozšířenou realitu za pomocí herního enginu Unity, **je ideální volbou pro vývojáře, kteří požadují jednoduchost** a chtějí se věnovat primárně implementaci svého řešení a neztrácet čas opakovanou práci s integraci funkcí hardware. Programátoři, kteří jsou ostřílení v editoru Unity mohou začít vyvíjet AR aplikace téměř ihned, případně jednoduše migrovat své Unity projekty do AR. Pro ty, kteří Unity neznají je seznámení s editorem otázka několika minut a díky naprosto vynikající dokumentaci ze strany vývojářů, je výuka velmi jednoduchá.

Velmi se mi líbí **flexibilní práce se scénou, strukturování jednotlivých objektu** a přirozená specifikace jejich atributu. Vývojáři, kteří znají koncept OOP budou jako doma. Provázanost s IDE Visual Studio představuje pro .NET vývojáře, kteří nejsou ostřílení v editoru Unity velmi známe prostředí.

AR funkcionalitu si může každý vývojář řešit individuálně, je však možné užití balíku **Mixed Reality Tookit**, který dovoluje implementovat jednotlivé AR prvky téměř způsobem Drag and Drop. Poskytuje jak integraci funkcionality zařízení, tak předem definované UI ovládací prvky. Otevřenost balíku dovoluje modifikaci všech poskytnutých komponent a případné přizpůsobení pro více specifickou implementaci není žádný problém. Integrováním balíku Vuforia ve verzi 3.2018 nativně přímo do editoru Unity bylo navíc podstatně ulehčeno integrovaní rozpoznávání kotevních bodů, objektů a tvarů ve scéně.

8 Závěr

V této diplomové práci jsem se snažil čtenářům přiblížit koncept, historii a vývoj aplikací pro rozšířenou realitu se zaměřením na ekosystém společnosti Microsoft - Windows Mixed Reality. Popsal jsem základní rozdělení aplikací a obecné předpoklady hardware pro práci s rozšířenou realitou. V části věnované seznámení s vývojem aplikací jsem čtenáři popsal základní pilíře návrhu aplikace, představil vývojové prostředí Unity pro Windows Mixed Reality a přední AR balík Vuforia.

Vzhledem k tomu, že koncept práce spadá do zaměření AZD, v praktické části práce jsem se rozhodl implementovat rozdílné vizualizace dat s důrazem na využití funkcionality nabízené právě rozšířenou realitou - snímání scény, kotevních bodů a ručních gest.

Přesto, že vývoj pro ekosystém WMR byl v poslední době velmi náročný hlavně z hlediska neustálenosti platformy a vyžadoval studium mnoha rozdílných zdrojů a literatury, si troufám tvrdit, že výsledná aplikace dobře demonstruje užití konceptu AR a hardware Microsoft Hololens v oboru analýzy a vizualizace dat a představuje jasný koncept toho, jak by rozsáhlejší analýza dat na výkonnějším AR hardware mohla v budoucnu vypadat.

Koncept **rozšířené reality** je z velké části stále neprobádanou oblastí informatiky a s jeho existencí je seznámeno jen velmi úzké publikum. V osobním životě AR nemusí představovat pouhý futuristický koncept používání výpočetní techniky, nebo další formu zábavy, ale může být také neocenitelným nástrojem pro osoby s fyzickým postižením či jiným zdravotním problémem, kteří se potýkají s každodenními překážkami.

Přesto, že se stále nepochybně potýkáme s problémem standardizace při vývoji a návrhu AR aplikací, tak z mého pohledu není pochyb o tom, že rozšířená realita bude dalším obrovským odvětvím, který se dostane do popředí světa výpočetní techniky. Se slibovaným příchodem lepšího dedikovaného AR hardware, finalizace konceptů návrhu, vývojových nástrojů a osvětou veřejnosti o tom co vlastně augumented reality computing představuje, má AR obrovský potenciál stát v přední linii oborů táhnoucí inovaci primárně v oblasti průmyslu, ale také v řádu osobních mobilních zařízení.

Práce na diplomové práci byla pro mě osobně velmi velkým přínosem hlavně z hlediska seznámení se spoustou nových informací a práci s nástroji se kterými jsem neměl žádnou předchozí zkušenost. Velmi dobře jsem se naučil nejen základy samotného konceptu AR, návrhu aplikací podle doporučení předních vývojářů, udělal si jasnou vizi o stavu vývojových nástrojů na poli rozšířené reality, ale také jsem poprvé pracoval s herním enginem Unity - který je naprosto neocenitelným nástrojem pro vývoj širokého spektra graficky bohatých aplikací. Při vývoji jsem měl možnost pracovat s hardwarem Microsoft Hololens, studovat jeho architekturu a probádat všechny možnosti tohoto jedinečného zařízení. Téma práce bylo nejen ojedinělé, ale samotné studium konceptu rozšířené reality bylo hluboce zajímavé a osobně se mi jeví jako neocenitelná zkušenost do budoucna.

Literatura

[1] Augmented Reality (AR) - Statistics And Facts *Statista.com* [online]. 2018 [cit. 4.4.2019]. Dostupné z: https://www.statista.com/topics/3286/augmented-reality-ar/

[2] Sean Ong: *Beginning Windows Mixed Reality Programming: For HoloLens and Mixed Reality Headsets*. Apress, 2017. ISBN 978-1-4842-2769-5.

[3] Joshua Newham: *Microsoft HoloLens By Example*. Packt Publishing, 2017. ISBN 978-1-7871-2626-8.

[4] Matt Smith: *Unity 2018 Cookbook*. Packt Publishing, 2017. ISBN 978-1-7884-7190-9.

[5] Arth, Clemens and Grasset, Raphael and Gruber, Lukas and Langlotz, Tobias and Mulloni, Alessandro and Wagner, Daniel: *The History of Mobile Augmented Reality*. Dostupné z: https://www.researchgate.net/publication/275974448

[6] Matt Smith, Krystian Babilinski: *Augmented Reality for Developers: Build practical augmented reality applications with Unity, ARCore, ARKit, and Vuforia*. Packt Publishing, 2017. ISBN 978-1-7872-8643-6.

[7] What is Augmented Reality? – Types of AR and Future of Augmented Reality *(Hemendra Singh)* [online]. 2018 [cit. 19.2.2018]. Dostupné z: https://dev.to/theninehertz/what-is-augmented-reality–types-of-ar-and-future-of-augmented-reality–1en0

[8] Mixed Reality Academy: code, tutorials, and lessons *(Microsoft)* [online]. 2018 [cit. 19.2.2019]. Dostupné z: https://docs.microsoft.com/en-us/windows/mixed-reality/academy

[9] Vuforia Fusion *(Vuforia)* [online]. 2018 [cit. 14.2.2019]. Dostupné z: https://library.vuforia.com/articles/Training/vuforia-fusion-article.html

[10] Getting Started with Vuforia Engine in Unity *(Vuforia)* [online]. 2018 [cit. 12.2.2019]. Dostupné z: https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html

[11] Comparing Google ARCore and Apple ARKit *(Vieyra Software)* [online]. 2018 [cit. 4.2.2018]. Dostupné z: https://medium.com/@vieyrasoftware/comparing-google-arcore-and-apple-arkit-81b4727132ad

[12] Are there any limitations in Vuforia compared to ARCore and ARKit? *(Stackoverflow)* [online]. 2018 [cit. 4.2.2018]. Dostupné z: https://stackoverflow.com/questions/50811770/are-there-any-limitations-in-vuforia-compared-to-arcore-and-arkit/50816392

[13] ARToolKit History, Feature List,Introduction *(Washington HITlab - Philip Lamb and Team)* [online]. 2004 [cit. 4.2.2018]. Dostupné z: http://www.hitl.washington.edu/artoolkit/documentation/index.html

[14] Augmented Reality Design Guidelines *(Google)* [online]. 2018 [cit. 21.1.2018]. Dostupné z: https://designguidelines.withgoogle.com/ar-design/augmented-reality-design-guidelines/introduction.html

[15] Learn Level Design: Unity Basics in Under 2 Hours *(Udemy)* [online]. 2017-2018 [cit. 2.2.2018]. Dostupné z: https://www.udemy.com/learn-level-design-unity-basics-in-under-2-hours/

[16] Unity, Windows Mixed Reality Overview: Quick start guide *(Unity)* [online]. 2017-2018 [cit. 4.2.2018]. Dostupné z: https://docs.unity3d.com/2018.3/Documentation/Manual/wmrsdkoverview.html

[17] Open-Source Building Blocks for Windows Mixed Reality Experiences *(Dong Yoon Park (Microsoft Design))* [online]. 2017-2018 [cit. 2.2.2018]. Dostupné z: https://medium.com/@dongyoonpark/open-source-building-blocks-for-windows-mixed-reality-experiences-hololens-mixedrealitytoolkit-28a0a16ebb61

[18] Microsoft HoloLens: HPU Architecture, Detailed *(Chris Angelini)* [online]. 2017-2018 [cit. 14.2.2018]. Dostupné z: https://www.tomshardware.com/news/microsoft-hololens-hpu-architecure-28nm,32586.html

[19] HoloLens–The Path to 60fps *(Pete Daukintis)* [online]. 2017-2018 [cit. 23.4.2018]. Dostupné z: https://peted.azurewebsites.net/hololensthe-path-to-60fps/

[20] Beginning Data Visualization *(mbs278)* [online]. 2017-2018 [cit. 13.2.2018]. Dostupné z: https://sites.psu.edu/bdssblog/2017/04/06/basic-data-visualization-in-unity-scatterplot-creation/

[21] Lightweight CSV reader for Unity *(Teemu Ikonen)* [online]. 2017-2018 [cit. 4.2.2018]. Dostupné z: https://bravenewmethod.com/2014/09/13/lightweight-csv-reader-for-unity/

[22] Vuforia - Object Recognition *(Vuforia)* [online]. 2017-2018 [cit. 4.4.2018]. Dostupné z: https://library.vuforia.com/articles/Training/Object-Recognition

[23] Adjacency List Implementation in C Sharp *(http://theoryofprogramming.com)* [online]. 2017-2018 [cit. 10.4.2018]. Dostupné z: http://http://theoryofprogramming.com/adjacency-list-implementation-in-c-sharp/

[24] Understanding the Gaze and Adding a Gaze Input Cursor into your Unity 3D Holographic App *(Abhijit Jana)* [online]. 2017-2018 [cit. 13.4.2018]. Dostupné z: https://abhijitjana.net/2016/05/19/adding-a-gaze-input-cursor-to-your-unity-3d-holographic-app/

[25] Use Spatial Understanding to Query your Room with HoloLens *(Sebastian Gambolati)* [online]. 2017-2018 [cit. 4.2.2018]. Dostupné z: https://medium.com/southworks/how-to-use-spatial-understanding-to-query-your-room-with-hololens-4a6192831a6f

[26] Tutorial to use Buttons, Dialogs and more with MRDesignLab (HoloToolkit) *(Bruno Capuano)* [online]. 2017-2018 [cit. 12.1.2019]. Dostupné z: https://elbruno.com/2017/08/02/hololens-tutorial-to-use-buttons-dialogs-and-more-with-mrdesignlab-holotoolkit/

[27] MixedRealityToolkit-Unity Wiki *(Microsoft)* [online]. 2019 [cit. 4.4.2019]. Dostupné z: https://github.com/Microsoft/MixedRealityToolkit-Unity/wiki

[28] Model Target Generator User Guide *(Vuforia)* [online]. 2019 [cit. 14.4.2019]. Dostupné z: https://library.vuforia.com/articles/Solution/model-target-generator-user-guide.html

[29] Unity ARKit By Example *(John Tucker)* [online]. 2019 [cit. 4.4.2019]. Dostupné z: https://medium.com/@johntucker/unity-arkit-by-example-part-1-64ec079efdc5

Příloha

Na přiloženém CD se nachází praktická část a dodatky této diplomové práce. Organizace souborů na disku je následující:

- **Src -** Unity projekty praktické části diplomové práce (Požaduje minimálně Visual Studio 2017, Unity verze 2018.3.12, Windows 10 SDK verze 10.0.20240.0 a vyšší, Vuforia Engine 8.1.7 a vyšší)

  Aplikace využívá **Microsoft Mixed Reality Toolkit v2.0.0 RC1** a může požadovat importování nové verze MRTK balíku v případě budoucích změn.

- **Anchors -** Zdrojové soubory ke kotevním bodům (3D CAD Modely, Vuforia datasety, zdrojové obrázky QR kodu).

- **Xcode -** Testovací aplikace pro iOS (iPhone s iOS 11+)

- **Media -** Kolekce videomateriálu prezentující vizualizace na Hololens a Apple iPhone.

- **Koncept -** Nákresy návrhů aplikace.

Některé přiložené soubory jsou komprimovány vzhledem k větší velikosti.